T.C

KIRIKKALE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

SMS MESAJLARINI AYIRT ETMEDE METİN MADENCİLİĞİ TEKNİKLERİNİN UYGULANMASI VE BUNA GÖRE YAPAY ZEKA MODELİNİN GELİŞTİRİLMESİ

PROJE-1

Hazırlayan

Yağız YAMAN

Danışman

Dr. Öğr. Üyesi Fahrettin HORASAN

Ocak-2021

KIRIKKALE

ÖN SÖZ

Proje çalışmasında metin madenciliği ve doğal dil işleme teknikleri uygulanarak cep telefonuna gelen mesajların spam mesaj mı yoksa ham mesaj mı olduğunun çeşitli yapay zeka algoritmalarına göre modellenip, farklı modellerin başarı oranları kıyaslanmış, incelenmiştir.

Öncelikle çalışma konusunun belirlenmesinde ve çalışmanın hazırlanma sürecinin her aşamasında istediklerimi göz önünde bulundurup, bilgilerini esirgemeyen, beni farklı konularla da tanıştıran ve her daim elinden geldiğince bana yardımcı olan danışmanım Dr. Öğr. Üyesi Fahrettin HORASAN ‘ a teşekkürlerimi sunarım.

ÖZET

Yaman, Yağız, “Sms Mesajlarını Ayırt Etmede Metin Madenciliği Tekniklerinin Uygulanması Ve Buna Göre Yapay Zeka Modelinin Geliştirilmesi”, Proje-1, Kırıkkale, 2021.

Günümüzde mevcut veri tabanlarında bulunan ham verilerin her geçen gün artması, ham verilerden elde edilmek istenen bilgilerin de doğru ve güvenilir olma ihtiyacını da arttırmıştır. Bu nedenle veri madenciliği önemli bir çalışma alanı haline gelmiştir. Veri madenciliği ile elde bulunan sayısal haldeki verilerin analizi rahatlıkla yapılabilmekteyken, metin halde bulunan verilerin analiz edilmesi de önemli bir ihtiyaç halinde gelmiş ve metin madenciliği konusunda yapılan çalışmaları da artmıştır. Metinsel verilerin sayısallaştırılarak veri madenciliği algoritmalarına girdi oluşturabilecek hale dönüşmesini sağlayan metin madenciliği günümüzde büyük önem teşkil etmektedir.

Çalışmada metin madenciliği ve doğal dil işleme teknikleri uygulanarak cep telefonuna gelen mesajların spam mesaj mı yoksa ham mesaj mı olduğunun çeşitli yapay zeka algoritmalarına göre modellenip tahmini yapılmıştır. Bu tahmin yapılırken SMSSpamCollection veri seti kullanılıp önce veri setindeki veriler işlenerek; büyük-küçük harf dönüşümü, sayıların ayıklanması, noktalama işaretlerinin ayıklanması, stopwords (tek başına anlamı olmayan kelimeler) ayıklanması ve steamming (kelimeyi köküne indirgeme) işlemi gibi veri ön işleme aşamaları tamamlanmıştır. Daha sonra bu temizlenen veri setinin kendi içerisindeki veriler kullanılarak mesajların uzunluğu özniteliği elde edilmiş, eski (ön işlenmemiş) veri setinin mesaj uzunlukları ile kıyaslanmıştır. Sonrasında yine veri seti içerisindeki kategorik veriler sayılarla ifade edilmiş, vektörizasyon işlemi yapılmış ve makine öğrenmesi için kullanılmaya hazırlanmıştır. Makine öğrenmesi aşamasında öncelikle model oluşturularak elimizdeki ayıklanmış, temiz veri setini eğitim ve test olarak %20 oranında bölüp, sınıflandırma ve regresyon algoritmalarına ait toplamda 6 farklı algoritma modeli (K-Nearest Neighbors, Logistic Regression, Support Vector Machines (SVMs), Decision Trees, Random Forests, Navie Bayes) kullanılarak hangi modelin daha başarılı olduğu gözlemlenmiştir. Çalışma ve gözlem sonucunda en başarılı algoritmanın %98.6 başarı oranı ile Random Forests olduğu, yapılan özellik çıkarımı ve ön işlemenin başarı oranının artmasında etkili olduğu ortaya çıkmıştır.

Anahtar Kelimeler: Veri Madenciliği, Metin Madenciliği, Doğal Dil İşleme, Yapay Zeka, Makine Öğrenmesi

ŞEKİLLER

İÇİNDEKİLER

GİRİŞ

Dijital çağ ile beraber iş dünyasında, bilimde, filolojide ve daha pek çok alanda insanlar teknolojik desteğe daha çok önem verme eğilimindedir. Yapay zeka bu teknolojik desteklerin başını çekmekte neredeyse her alanda kendine yer bulan bir alan olmaya başlamıştır.

Yapay zeka (AI), makinelerin deneyimden öğrenmesini, yeni girdilere uyum sağlamasını ve insan benzeri görevleri gerçekleştirmesini mümkün kılar. Bugün duyduğunuz çoğu AI örneği - satranç oynayan bilgisayarlardan kendi kendine giden arabalara kadar - derin öğrenme ve doğal dil işlemeye dayanmaktadır. Bu teknolojileri kullanarak bilgisayarlar, büyük miktarda veri işleyerek ve verilerdeki kalıpları tanıyarak belirli görevleri yerine getirecek şekilde eğitilebilir.

Yapay zeka ve makine öğreniminden yola çıkarak geliştirilen veri madenciliği, bugün birçok sektördeki işletmeler için satışları artırmanın en iyi yollarından biri konumundadır. Veri madenciliği, istatistik, veri tabanı yönetim sistemleri ve makine öğrenmesi gibi birçok disiplini kullanarak daha önce keşfedilmemiş ve açık bir şekilde ortada olmayan bilgiyi çıkarmak için kullanılan veri analiz metodudur. Büyük bir veri yığınından yararlı bilgiyi çekip, yani veriden bilgi çıkarabilmek oldukça zahmetli bir iştir. Madencilik sonucunda edinilen kazanımları göz önünde bulundurursak şirketler için sadece sahip oldukları verileri değil dışarıdan alınan verileri de koruyabilmek ve işleyebilmek son derece hassas bir konu haline gelmiştir. Bu nedenle veri madenciliği her geçen gün önem kazanmakta ve ihtiyaç haline gelen bir alan olmaya başlamıştır. Veri madenciliği veri tabanlarındaki sayısal halde bulunan verilerin çeşitli istatistiksel, analitik yöntemlerle analiz edilmesi ve elde edilen sonuçların yorumlanması ile ilgilenir. Fakat sayısal halde bulunmayan verilerin analiz edilmesi ihtiyacı sonucu metin halinde bulunan verilerin analizi hususunda da çeşitli çalışmalar yapılması gerekliliği duyulmuş ve sonuçta metin madenciliği alanı oluşmuştur. Metin madenciliği günümüzde kullanılan fakat çok yeni bir alandır.

Metin madenciliği çalışmaları metni veri kaynağı olarak kabul eden veri madenciliği çalışmasıdır. Metin üzerinden yapısallaştırılmış veri elde etmeyi amaçlar. Metin madenciliği çalışmaları, veri madenciliğinin bir parçası olarak düşünülmesine rağmen, alışılagelen veri madenciliğinden farklıdır. Ana farklılık, Metin madenciliğinde örüntülerin olay tabanlı veri tabanlarından daha çok, doğal dil metinlerinden çıkarılmasıdır. Metin madenciliği çalışmaları, metin kaynaklı literatürdeki diğer bir çalışma alanı olan doğal dil işleme (natural language

processing, NLP) çalışmaları ile çoğu zaman beraber yol yürümektedir. Doğal dil işleme çalışmaları daha çok yapay zeka altındaki dil bilim bilgisine dayalı çalışmalarını kapsamaktadır. Metin madenciliği çalışmaları ise daha çok istatistiksel olarak metin üzerinden sonuçlara ulaşmayı hedefler. Metin madenciliği çalışmaları sırasında çoğu zaman doğal dil işleme kullanılarak özellik çıkarımı da yapılmaktadır.

BİRİNCİ BÖLÜM

VERİ MADENCİLİĞİNİN TEORİK YAPISI

1.1 VERİ MADENCİLİĞİ NEDİR?

Veri madenciliği büyük ölçekli yığın haldeki veriler arasından “işe yarayan”, “değerli olan” bir bilgiyi bu büyük yığından elde etme sürecidir. Veri madenciliği verilerin, belirli aşamalardan geçmesi sonucunda veriden bilgiye dönüşümüdür. Bu kapsamda veri madenciliği çok değerli ve her geçen gün gelişen dijital çağda kendine yer bulabilen önemli bir alandır. Veri madenciliğinin amacı; temel ve basit olarak uygulandığı alana göre veriler üzerinden çıkarım yapmak, veriden ileriye dönük fikirler edinmek ve buna göre tavsiye sistemleri planlaması yapılarak ileriye yönelik yol haritası oluşturmaktır.

Veri madenciliği bu büyük veri yığınları arasından “işe yarar” diyebileceğimiz veriyi, yani bilgiyi çıkartmak için istatistik ve yapay zeka tabanlı çok sayıda yöntem, algoritma ve uygulama kullanmaktadır. Veri madenciliği algoritmaları; istatistiksel algoritmalar, matematiksel algoritmalar ve yapay zeka algoritmalarını bir arada içerir.

Veri madenciliği veriden bilgiye ulaşma sürecinde çok büyük veriler kullanmakla birlikte, veri setleri dediğimiz; araştırılan konuya ya da olguya özgü etraflıca ve dikkatlice toplanmış boyutları değişken olabilen veri topluluklarını da kullanabilir.

Veri madenciliğine başlarken problemin tanınması, ne için veri madenciliği yapılmak istenmesi gibi durumlar göz önüne getirilerek buna uygun veri seti toplanması ile özel bir konuda veri madenciliği yapılabilmektedir. Bir veri madencisi, karşısındaki probleme göre hangi algoritmayı, hangi yöntemi veya hangi istatiksel teoriyi kullanacağını araştıran, hangi algoritmanın, yöntemin ve teorinin daha başarılı sonuçlar vereceğini gözlemleyen ve başarı oranına göre bu olayları tekrar edip etmeme kararını alan kişidir. Örneğin; iris çiçeğinin taç ve çanak yapraklarının enlerinin ve boylarının uzunluğuna göre içinde barındırdığı 3 türden (Virginica, Versicolor, Setosa) hangisi olduğuna karar verilmektedir. Buradaki yaprakların en boy parametreleri karar üzerinden doğrudan etkili olup bu parametrelere göre tahminler gerçekleştirilmektedir. Bu ve bunun gibi özel problemler için özel algoritmalar veya özel modellemeler kullanılabilir.

Bu çalışmamda benim ele aldığım problem; veri madenciliği, metin madenciliği ve makine öğrenmesi disiplinleri doğrultusunda makineye telefonlarımıza gelen mesajın etiket adı tahminini yaptırmaktır. Bu etiket adı kullandığım veri setinde 2 (ham, spam) adettir. Spam mesajlar; isteğimiz dışında ve genellikle toplu bir şekilde telefonlarımıza ya da maillerimize gelen mesajlardır. Genellikle dikkat çekici olup, reklam amaçlı olabileceği gibi kötü amaçlı olanları da vardır. Ham mesaj dediğimiz terim ise; genellikle istenen ve spam olarak görülmeyen mesaj türüdür.

Benim bu çalışmada kullandığım veri seti SMSCollection; 5572 satır ve 2 sütundan oluşmaktadır. 5572 satır içerisinden 4825 tanesi ham mesaj, 747 tanesi spam mesajdır. Sütunlar da ham ve spam olmak üzere iki adettir. Bu veri setini SMSCollection adı altında web üzerinde rahatlıkla bulabileceğiniz; telefonlarımıza gelen kısa mesajların, makine tarafından spam mesaj olup olmadığının tahminini yapması amacı ile toplanmıştır. Bu veri setinin toplanmasında emeği geçenler:

- Grumbletext Web sitesinden 425 SMS spam mesajından oluşan bir koleksiyon manuel olarak çıkarılmıştır. Bu, cep telefonu kullanıcılarının SMS spam mesajları hakkında kamuya açık iddialarda bulundukları bir İngiltere forumudur.
- Singapur Ulusal Üniversitesi Bilgisayar Bilimleri Bölümü'nde araştırma için toplanan yaklaşık 10.000 meşru mesajdan oluşan bir veri kümesi olan NUS SMS Corpus'un (NSC) rastgele seçilmiş 3,375 SMS ‘inden oluşan bir alt kümesidir. Mesajlar büyük ölçüde Singapurlulardan ve çoğunlukla Üniversiteye devam eden öğrencilerden geliyor. Bu mesajlar, katkılarının kamuya açıklanacağı konusunda bilgilendirilen gönüllülerden toplandı.
- Caroline Tag'ın Doktora Tezinden toplanan 450 SMS mesajının bir listesi,
- Ve SMS Spam Corpus’ un v.0.1 Big'i dâhil edilmiştir. Bu, 1.002 SMS ve 322 spam mesajından oluşmaktadır.

Bu külliyat (veri seti) aşağıdaki akademik araştırmalarda da kullanılmıştır:

- Content Based SMS Spam Filtering. 2006 ACM Belge Mühendisliği Sempozyumu Bildirileri (ACM DOCENG'06), Amsterdam, Hollanda, 10-13, 2006.
- Mobil (SMS) spam filtreleme. Bilgi Erişiminde Araştırma ve Geliştirme üzerine 30. Yıllık uluslararası ACM Konferansı Bildirileri (ACM SIGIR'07), New York, NY, 871-872, 2007.
- Kısa mesajlar için spam filtreleme. Bilgi ve Bilgi Yönetimi üzerine 16. ACM Konferansı Bildirileri (ACM CIKM'07). Lizbon, Portekiz, 313-320, 2007.

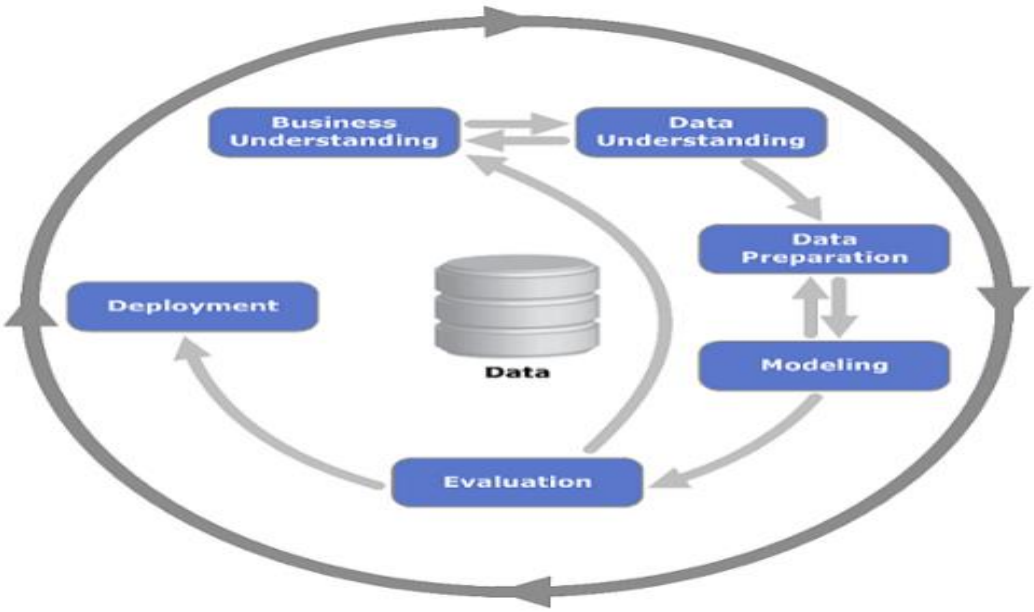
İKİNCİ BÖLÜM

VERİ MADENCİLİĞİ YAKLAŞIMLARI

2.1. VERİ MADENCİLİĞİ METODOJİLERİ

“ Metodolojiler, önemli veri madenciliği sorunlarını daha iyi anlamak ve süreçlerinin nasıl yapılması gerektiğini ifade eden yöntemlerdir. Bu metodolojilerden en çok kullanılan CRISP-DM ve SEMMA metodolojileri hakkında genel olarak bilgi vermeye çalışacağım. Bu metodolojiler dışında ise şirketlerin kullandığı kendine özel metodolojileri de bulunmaktadır.

- **CRIPS-DM Metodolojisi (Cross-Industry Standard Process for Data Mining)**



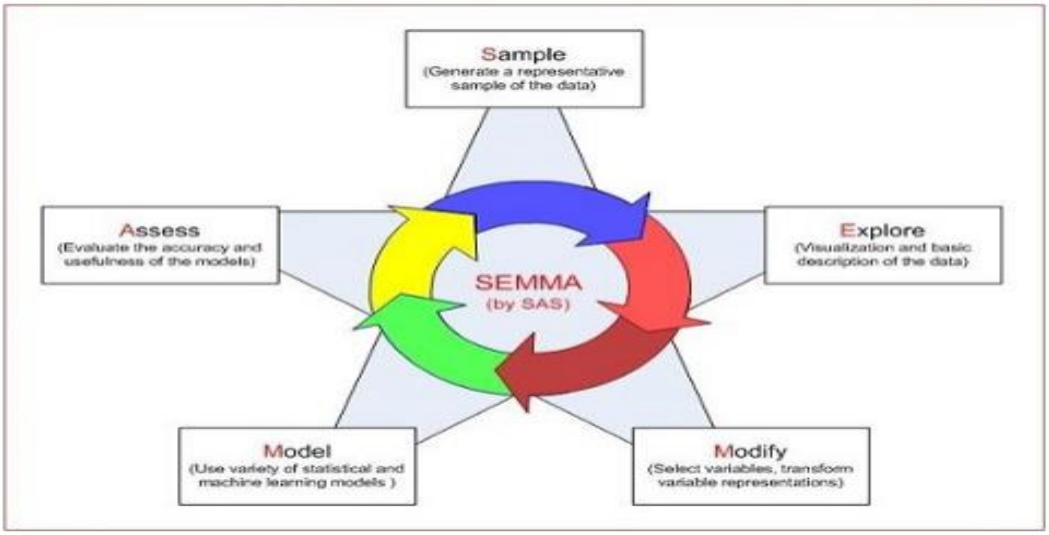
Şekil 1. CRIPS-DM

CRIPS-DM analitik, veri madenciliği ve veri biliminde en popüler metodolojidir. Veri madenciliği projelerini planlama ve yürütmede kullanılan bir süreç modelidir. Bu model 6 aşamadan oluşmaktadır.

1. **İşi Tanımlama (Business Understanding):** Başlangıç olarak proje hedeflerini ve ihtiyaçlarını anlama ve bunu veri madenciliği tanımına dönüştürme aşamasıdır.

2. **Veriyi Anlama (Data Understanding):** Bu aşamada veri toplama işlemiyle başlar, veri kalitesi problemlerini belirleme, veriden ilk görüleri çıkartma… diye verinin probleme ne kadar çözüm getirdiğiyle devam eder.
3. **Veriyi Hazırlama (Data Preparation):** Topladığımız veriden veri seçme, veri temizleme, veri dönüştürme… gibi model uygun son veri setini elde etmek için yapılan işlemlerdir.
4. **Modelleme (Modeling):** Bu aşamada çeşitli modelleme tekniklerinin ve algoritmalarının seçilmesi, parametrelerin seçilmesi ve uygulama işlemleri gerçekleştirilir.
5. **Değerlendirme:** Bu aşamada oluşturulan modelin deneme ve gözden geçirilmesi yapılır, gerekiyorsa iyileştirmeler yapılır.
6. **Uygulama:** Son aşamada ise modelin analistlere ve son kullanıcılara sunulup iş süreçlerinde kullanılacak hale getirilir.

- **SEMMA Metodolojisi (Sample, Explore, Modify, Model and Assess)**



Şekil 2. SEMMA

Sample, Explore, Modify, Model ve Assess kelimelerinin baş harfleri ile oluşan bir metodolojidir. İstatistik ve İş Zekası yazılımı geliştiren SAS Enstitüsü tarafından geliştirilen ardışık adımlar listesidir. CRISP-DM' den farkı ise CRISP-DM olduğu

gibi bütün projenin metodolojisi iken SEMMA ise veri madenciliği yapılan kısmın metodolojisidir.

1. **Sample:** Bu aşama veri örnekleme ile başlar yani modelleme için veri seti seçilir.
2. **Explore:** Beklenen ve beklenmeyen değişkenler arasında ilişkileri ve anormallikleri keşfedilerek verilerin anlaşılır hale getirilir
3. **Modify:** Modelleme süreci için verilerin temizlenmesi ve dönüştürülmesi yapılır.
4. **Model:** Eğilim ve tahminleri keşfetmek için modelin verilere uygulanmasıdır.
5. **Assess:** Bu aşamada uyguladığımız modelin sonucumuza uygunluğunun değerlendirilmesi yapılır. ” [1]

[1] Furkan Alaybeg, Veri Madenciliği Giriş, Yöntemleri ve Metodojileri, Medium, Haziran 24, 2019, https://medium.com/@furkanalaybeg/veri-madencili%C4%9Fi-ve-y%C3%B6ntemleri-d0e2fd238e44

ÜÇÜNCÜ BÖLÜM

VERİ MADENCİLİĞİ TEKNİKLERİ

3.1. VERİ MADENCİLİĞİ SÜREÇLERİ

Veri madenciliğinin yol haritası bu süreçleri tanımaktan, işlemekten ve kontrol etmekten bazen ise bunların hepsini tekrar tekrar yapmaktan geçmektedir. Bu süreçler aslında yukarıda metodolojilerde anlatılan her bir başlığı aslında son derece önemli olan, başarılı bir veri işleme yapabilmek için olmazsa olmazlardandır. Bu çalışmamda da bu süreçleri eksik ve tam uygulamış değerlendirme ve sonuç kısmında burada yaptığım çalışmaların etkisi fazlasıyla görmekteyim.

Şimdi aşağıda bu süreçleri daha yakından inceleyerek çalışmamda bu aşamalarda ne gibi zorluklarla karşılaştığımı ve bunlara nasıl çözümler bulduğumu, hangi teknikleri kullandığımı sırası ile anlatacağım. Öncelikle anlatacağım bütün teknikler Anaconda Navigator üzerinden Jupyter Notebook ara yüzü ile Python dilinde yazılmış kodlar ile gerçekleşmiştir.

3.1.1 Problemin Tanımlanması

Problemi tanımak veri madenciliği süreçleri arasında en başta ve kesin, net bir şekilde yapılması gereken bir unsurdur. Problem belirlenmeden geriye kalan süreçlerin bir anlamı olmamakla beraber sonuçların ve modellerin doğruluk oranları son derece alakasız olabilir.

Çalışmamda ise problem; telefonlara gelen kısa mesajın (SMS) spam ise spam mesaj, ham ise ham mesaj şeklinde kategorik tahminler yapan bir makine öğrenmesi modeli eğitmek ve bunu geliştirmek. Bu şekilde problemi tanımladıktan sonra sırada bu problemle alakalı verilere göz atmamız gerekmektedir.

3.1.2. Verinin Tanınması ve Veri Ön İşleme

Veriyi tanımak, veri hakkında yorum yapabilmektir. Birden fazla verilerin toplanıp bir yığın, topluluk oluşturması bize veri setini ifade etmekteydi. Bu çalışma için örnek verecek olursak; çalışmada kullanılan veriler metinsel veriler olduğundan aslında veriyi metin olarak kabul eden bir veri madenciliği türü olan metin madenciliğinde verinin tanınması ve hazırlanması belki de en fazla zaman alan ve en dikkat edilmesi gereken önemli bir iştir. Özellikle metin madenciliği yaparken veri setlerini incelendikçe içerisinde bulundurdukları ham verilerin; aykırı verilerden,

veriler arasına karışmış rakamlardan, noktalama veya başka karakter işaretlerinden, büyük küçük harflerden, tek başına anlam ifade etmeyen kelimelerden (stopwords) ve ek almış kök durumunda olmayan kelimelerden oluştuğu gözümüze çarpacaktır. Bu durumda yapmamız gereken veriyi hazırlama işlemi problemimize göre değişkenlik gösterebilir. Eğer problemimizde belli bir karakter adeti veya özel bir karaktere göre tarama ya da analiz yapacaksak bütün karakterleri veri setinden çıkarmak işimize gelmeyebilir. Başka bir önek; çalışmada noktalama işaretlerinin sayısının bizim için önemi var ise önce ham veriler üzerinden özellik çıkarımı (Feature Extraction) yapabilir ve daha sonra noktalama işaretlerini temizleyebiliriz. Yani her problem için veriyi hazırlama, başka bir ifade ile veri ön işleme (Preprocessing) işlemleri değişkenlik gösterebilir.

Bu çalışmada işimiz mesajlar ile ilgili olduğu bizim için önemli olan her spam mesajın belirgin özellikleri arasında olan mesaj uzunluğu, noktalama işaretleri sıklığı, sayıların kelimeler arasına girmesi gibi etkenler bizim için önemli ve ayırt edici olacaktır. Bu sayı ve noktalama işaretleri kaldırıldığında ortada çok anlamsız bir mesaj kalacağından bu mesajın spam olduğu açıkça belli olacaktır. Bu kapsamda öncelikle ham veri setimizdeki karakterlerin uzunluğunu buluyoruz. Bu değer ile daha sonra temizlenmiş, işlenmiş mesajların uzunluk değerini karşılaştıracağız. Daha sonra az önce saydığımız; aykırı veriler, veriler arasına karışmış rakamlar, noktalama veya başka karakter işaretleri, büyük küçük harfler, tek başına anlam ifade etmeyen kelimeler (stopwords) ve ek almış kök durumunda olmayan kelimeler gibi verilerden kurtulup mesajımızı temiz bir hale getiriyoruz. Bu verilerden Python içerisinde var olan “nltk”, “re”, “stopwords” ve “PorterStemmer” kütüphaneleri, modülleri sayesinde kurtuluyoruz.

Bu modül ve kütüphaneleri inceleyecek olursak;

- Re (Regular Expressions | Düzenli ifadeler): Düzenli ifadeler tüm modern dillerde bulunur, dağınık bir metin içerisinde istediğimiz formattaki metinleri yakalayabilmemize imkân tanır. Mesela, bir kaynakta geçen tüm e-posta adreslerini veya içinde rakam bulunan ve gmail uzantılı olan mail adreslerini ayıklamak için kullanabilirsiniz. Düzenli ifadeler olmasaydı ardı arkasına birçok if-else kod blokları yazmak gerekebilirdi. Bu modül, birkaç

saatte yapabileceğiniz bir işlemi saniyeler içerisinde sizin yerinize yapabiliyor.

- Natural Language Toolkit (NLTK): İstatistiksel doğal dil işlemede (NLP) uygulamak için insan dili verileriyle çalışan Python programları oluşturmak için kullanılan bir platformdur. Simgeleştirme, ayrıştırma, sınıflandırma, saplama, etiketleme ve anlamsal akıl yürütme için metin işleme kitaplıkları içerir.
- Stopwords: Nltk ‘in corpus modülü altında bir fonksiyondur. Tek başına anlamı olmayan kelimelerin (the, a, with gibi) bize metin madenciliği ve doğal dil işlemede ileride kuracağımız algoritmalarımızda bir faydası yoktur, hatta anlamsal olarak bir şey ifade etmedikleri için silinmezlerse algoritma ve tahminlerin doğruluk oranlarında ciddi azalmalar görülebilir. Bu sebeple bu kelimelerin tespit edilip metinden (corpustan) çıkartılması gerekmektedir. Burada bu anlamsız kelimeler dillere göre değişiklik göstereceğinden kendi içerisinde seçebileceğimiz farklı dil paketleri mevcuttur.
- PorterStemmer: Bu aslında bir modülden çok nltk’ in stem veya porter modüllerinin altındaki bir fonksiyondur. Temel görevi 'Stemming' işlevidir. Peki bu işlev nedir?
    - Bir kelimeyi köküne indirgeme işlemi yapmaktadır.
    - Buradaki çalışmada İngilizce sözcükler ile çalıştığımız için kelimelerin sonundaki (-ed,-ity gibi) takıların gitmesini beklemekteyiz.

	message	labels	len	clean_message	clean_message_len
0	Go until jurong point, crazy.. Available only ...	0	111	go jurong point crazi avail bugi n great world...	76
1	Ok lar... Joking wif u oni...	0	29	ok lar joke wif u oni	21
2	Free entry in 2 a wkly comp to win FA Cup fina...	1	155	free entri wkli comp win fa cup final tkt st m...	101
3	U dun say so early hor... U c already then say...	0	49	u dun say earli hor u c alreadi say	35
4	Nah I don't think he goes to usf, he lives aro...	0	61	nah i think goe usf live around though	38

Şekil 3. Kıyaslama

Veri ön işleme sonucu eski ham veri ile yeni temiz veriler arasındaki farklar görülmektedir.

Bu aşamalar düzgün bir şekilde tamamlandığında veri ön işleme bitmiş ve artık veri makine öğrenmesi için biraz daha düzgün duruma getirilmiştir.

3.1.3. Model oluşturma ve Modeli Makine Öğrenmesine Hazırlama

Aslında bakarsak makine öğrenmesine kadar yapılan her işlem veriyi makine öğrenmesine hazırlama işlemidir. Fakat burada belirtilmek istenen durum artık veri üzerinde en son işlemlerin yapıldığı, makine öğrenme algoritmalarının sonucuna doğrudan etkili olacak ayarlamaların yapıldığı ve gerektiğinde bu ayarların değiştirildiği aşamadır.

Bu aşamada öncelikle verilerimizi vektöre etmeliyiz.

- **Vektörizasyon (Vectorization):** Aslında bu kavramın kök tanımı; birçok CPU'nun aynı işlemi iki, dört veya daha fazla veriye aynı anda uygulayan "vektör" veya "SIMD" komut setleri bulunur. Modern x86 yongalarında SSE yönergeleri bulunur, birçok PPC yongasında "Altivec" yönergeleri bulunur ve hatta bazı ARM yongalarında NEON adı verilen bir vektör komut kümesi bulunur. Burada bahsi geçen "SIMD","SSE" ve "Altivec" yönergelerinin temel amacı ve hizmeti birden fazla veri nesnesi üzerinde aynı işlemlerin gerçekleştirildiği performansı artıran işlemlerdir.

  Vektörizasyon, skaler bir programı vektör programına dönüştürmek için kullanılan terimdir. Vektörize edilmiş programlar, tek bir komuttan birden fazla işlemi çalıştırabilirken, skaler programlar yalnızca aynı anda işlenen çiftleri üzerinde çalışabilir.

  Yani kısacası "Vektörizasyon", bir dizinin tek bir öğesini N kez işlemek yerine, dizinin 4 öğesini aynı anda N / 4 kez işleyecek şekilde bir döngüyü yeniden yazma işlemidir.

  Vektörize kodun birçok avantajı vardır, bunlar arasında:

  - Vektörize edilmiş kod daha kısa ve okunması daha kolaydır.
  - Daha az kod satırı genellikle daha az hata anlamına gelir.
  - Vektörizasyon olmadan, kodumuz verimsiz ve döngüler için

Bizim burada Vectorization’ u kullanmamızın nedeni de yukarıdaki nedenler olmakla birlikte zaten hali hazırda kullandığımız Numpy, dizileri işlemek için yerleşik Python seçenekleriyle yapıldığında genellikle daha yavaş olan n boyutlu dizinin hızlı işlenmesi için vektörleştirmeyi kullanmaktadır. Bu sayede kodumuz ileride geliştireceğimiz makine öğrenmesi algoritmasını da darboğaza sokmaz ve hem işlemciyi hem de derleyici verimli halde kullanılırız.

Vektöre edilmiş veriler ileride eğitim ve test olarak model eğitimi için kullanılacağından ve makine öğrenmesinde tahminler yapılacağından etiketlerine göre ayrı ayrı vektörlere ayrılırlar. Kendi çalışmamda bu kısımda mesaj verileri ve etiket verileri ayrı X ve Y değişkenlerine atanmış durumdadır.

Sırada ise metin madenciliğinde önemi son derece büyük olan kelimelerin geçme sıklıkları ile ilgili bir işlem yapılacaktır. Bu işlem verilen bir metni özelliklerin geçme sıklığı matrisine çeviren bir işlemdir. Bu işlem Python’ da sklearn kütüphanesi altında bulunan fonksiyonlardan birisi olan CountVectorizer ile gerçekleştirilir. CountVectorizer’ in içerisinde son derece önemli bir parametre vardır; bu parametre kesinlikle isteğe bağlıdır ve eldeki veri setinin içeriğine göre değerlendirilip yazılmalıdır. Burada yaptığı işlem 5572 metin arasında kelime frekanslarına bakarak en verimli, en çok 2500 kelime frekansını alır. Yani bazen tüm kelime dağarcığını dönüştürmek etkili değildir, çünkü eğer son derece nadir (frekansı az) bulunan veriler geçilirse (dönüştürülürse) gelecekte girdilere istenmeyen boyutlar katacaktır. Bu durumda uygun tekniklerden biri, veri setindeki kelime frekanslarını yazdırmak ve daha sonra bunlar için belirli bir eşik belirlemek olabilir. 50'lik bir eşik belirlediğinizi ve veri topluluğunuzun 100 kelimeden oluştuğunu düşünün. Sözcük frekanslarına baktıktan sonra 20 sözcük, 50'den az kez geçiyorsa (frekansı). Böylece, max_features parametresini 80 olarak belirlemek modeliniz için daha iyi olacaktır.

Bu aşamadan sonra artık veriler model oluşturma ve daha sonra makine öğrenmesi için hazır duruma gelmiştir. Bu aşamada yapılan işlemler, makine öğrenmesi başarısına göre değiştirilip tekrar gözlemlenebilir. Bu nedenle bu aşama veri madencisi ve uğraştığı problem ile alakalı değişken bir aşamadır.

3.1.3.1 Model Oluşturma

Artık en baştaki veri seti üzerinde işlemler yapılmış, arındırılmış temiz bir veri setine dönüşümünü tamamladı ve model oluşturmak için uygun hale geldi. Bu durumda artık modelin eğitilmesi ve test edilmesi için modelimizde kullanacağımız bu verileri ölçeklemeye ihtiyacımız vardır. Bunun için yine Python' un sklearn kütüphanesinden fonksiyonları kullanmaktayız.

- **fit():** Eğitim verilerinden öğrenme modeli parametreleri oluşturmak için kullanılır. Verilerde bulunan özelliklerin her birinin ortalamasını ve varyansını hesaplar.
- **transform():** fit() yönteminden üretilen verilerin ilgili ortalama ve varyansı kullanarak tüm özellikleri dönüştürür.
- **fit_transform():** Eğitim verilerini ölçeklendirebilmek ve ayrıca bu verilerin ölçeklendirme parametrelerini öğrenebilmek için eğitim verilerinde fit_transform() kullanılır. Burada oluşturulan model, eğitim setinin özelliklerinin ortalamasını ve varyansını öğrenir. Öğrenilen bu parametreler daha sonra test verilerini ölçeklendirmek için kullanılır.

Burada bilinmesi gereken önemli bir husus vardır. fit_transform() fonksiyonunu eğitim verilerinde, transform() fonksiyonu ise test verilerinde kullanılır. Bunun nedeni eğitim verilerini ölçeklendirebilmek ve ayrıca bu verilerin ölçeklendirme parametrelerini öğrenebilmek için eğitim verilerinde fit_transform() kullanılır. Burada oluşturulan model, eğitim setinin özelliklerinin ortalamasını ve varyansını öğrenir. Öğrenilen bu parametreler daha sonra test verilerini ölçeklendirmek için kullanılır. Aslında bu fonksiyon kendi içerisinde; fit() fonsksiyonu, verilerde bulunan özelliklerin her birinin ortalamasını ve varyansını hesaplıyor. transform() fonksiyonu ise, ilgili ortalama ve varyansı kullanarak tüm özellikleri dönüştürüyor. Şimdi, ölçeklendirmenin test verilerimize de uygulanmasını istiyoruz ve aynı zamanda modelimizle önyargılı olmak istemiyoruz. Test verilerimizin tamamen yeni ve modelimiz için sürpriz bir set olmasını istiyoruz. transform() fonksiyonu bu durumda bize yardımcı olur. transform() fonksiyonu kullanarak, test verilerimizi dönüştürmek için eğitim verilerimizden hesaplananla aynı ortalama ve varyansı kullanabiliriz. Bu

nedenle, eğitim verilerini kullanarak modelimiz tarafından öğrenilen parametreler, test verilerimizi dönüştürmemize yardımcı olacaktır.

Yani biz eğer test verilerimizde de fit() fonksiyonu kullanırsak, her özellik için yeni bir ölçek olan yeni bir ortalama ve varyans hesaplayacağız ve modelimizin test verilerimiz hakkında bilgi edinmesine de izin vereceğiz. Bu nedenle, sürpriz olarak saklamak istediğimiz şey artık modelimiz tarafından bilinmekte olup, modelimizin makine öğrenimini kullanarak bir model oluşturmanın nihai amacı olan test (yani görünmeyen, bilinmeyen) verilerinde nasıl performans gösterdiğine dair iyi bir tahmin alamayacağız. Bu, bir makine öğrenimi modeli oluştururken verilerimizi ölçeklendirmek için kullanılan standart prosedürdür, böylece modelimiz veri kümesinin belirli bir özelliğine yönelik önyargılı olmaz ve aynı zamanda modelimizin test verilerimizin özelliklerini, değerlerini ve eğilimleri gibi parametreleri öğrenmesini engeller.

Şimdi ise bu oluşturulan modelden eğitim ve test için kullanacağımız verileri almamız ve daha sonra makine öğrenmesi aşamasında bu veriler yardımı ile bunları kullanmamız gerekmektedir. Bunun için sklearn' de train_test_split() fonksiyonu bulunmaktadır. Bu fonksiyon, içerisine yukarıda vektörleştirdiğimiz X ve Y ve değerlerini, test_size denilen verinin kaçta kaçını test için kullanacağımızı belirttiğimiz parametreyi ve random_state denilen bir parametreleri alır.

- **random_state:** Oluşturduğumuz bölmelerin yeniden üretilebilir olmasını sağlar. Scikit-learn, bölünmeleri oluşturmak için rastgele permütasyonlar kullanır. random_state, rastgele sayı oluşturucu için bir çekirdek olarak kullanılır. Bu, rastgele sayıların aynı sırada üretilmesini sağlar. Eğer kodda random_state belirtmezsek, kodumuzu her çalıştırdığımızda yeni bir rastgele değer üretilir ve eğitim ve test veri kümeleri her seferinde farklı değerlere sahip olur. Bununla birlikte, random_state = 0, 1, 42 veya başka bir tam sayı gibi sabit bir değer atanırsa, kodumuzu kaç kez çalıştırırsak çalıştıralım, sonuç aynı kalır ve dolayısı ile eğitim ve test veri kümelerinde aynı değerler olacaktır.
- **test_size:** Vektörize edilmiş ve ayrılmaya hazır verilerin kaçta kaçını eğitim ve test olarak ayırmamızı belirteceğimiz parametredir. Modelin başarısına göre tekrar tekrar değiştirilip gözlemlenebilmekle birlikte, genellikle %20-%33 arasında değerler almaktadır. Bu değerler veri setinin büyüklüğüne göre değişmekte olup, ayrılma oranına göre farklı sonuçlar kaşımıza sunabilir.

Bu aşamalardan sonra artık modelimiz makine öğrenmesi algoritmaları için hazır duruma gelmiştir. Şimdi bu algoritmaları modelimiz üzerinde uygulayıp, gözlemler yaparak sonuç üzerinde etkili olabilecek değişiklikler ile modelimizi geliştirmeliyiz.

3.1.4. Makine Öğrenmesi Yöntemlerinin Uygulanması ve Değerlendirme

"Makine öğrenmesi yöntemlerini denetimli ve denetimsiz olmak üzere iki ana kategoriye ayırmak mümkündür. Makine öğrenmesinde iyi tanımlanmış veya kesin bir hedef olduğunda denetimli (supervised) ifadesi kullanılır. Elde edilmesi istenen sonuç için özel bir tanımlama yapılmamışsa veya belirsizlik söz konusu ise denetimsiz (unsupervised) ifadesi kullanılır.

Denetimli ve denetimsiz ifadeleri birbirinin tersine karşılık gelmektedir. Denetimli ve denetimsiz yöntemleri sürecin bütünü açısından değerlendirmek gerekirse;

- Denetimsiz yöntemler daha çok veriyi anlamaya, tanımaya, keşfetmeye yönelik olarak kullanılan ve sonraki uygulanacak yöntemler için fikir vermeyi amaçlamaktadır,
- Denetimli yöntemler ise veriden bilgi ve sonuç çıkarmaya yönelik kullanılmaktadır, denilebilir. Bu nedenle denetimsiz bir yöntemle elde edilen bir bilgi veya sonucu, eğer mümkünse denetimli bir yöntemle teyit etmek, elde edilen bulguların doğruluğu ve geçerliliği açısından önem taşımaktadır.

Denetimli (Supervised) öğrenme yöntemleri:

- En yakın k komşuluk (K-Nearest Neighbors)
- Destek vektör makineleri (Support Vector Machines (SVMs))
- Regresyon modelleri (Regression Models)
- Rastgele Ormanlar (Random Forests)
- Karar ağaçları (Decision Trees)
- Navie Bayes

Denetimsiz (Unsupervised) öğrenme yöntemleri:

- Aşamalı kümeleme (Hierarchical clustering)
- Kendi kendini düzenleyen haritalar (Self organized maps)

olarak sınıflandırılabilir.

Veri madenciliği ve makine öğrenmesi ile ilgili kullanılan pek çok yöntemin yanına hemen her geçen gün yeni yöntem ve algoritmalar eklenmektedir. Bunlardan bir kısmı onlarca yıldır kullanılan klasik teknikler diyebileceğimiz ağırlıklı olarak istatistiksel yöntemlerdir. Diğer yöntemler de genellikle istatistiği temel alan ama daha çok makine öğrenimi ve yapay zekâ destekli yeni nesil yöntemlerdir.

Kullanılan klasik yöntemlerin başlıcaları;

- Regresyon,
- K - En Yakın Komşuluk,
- Kümeleme

olarak sayılabilir.

Yeni nesil yöntemlerin başlıcaları ise;

- Karar Ağaçları,
- Birliktelik Kuralları,
- Yapay Sinir Ağları,

olarak sıralanabilir.

Ayrıca diğer veri madenciliği ve makine öğrenmesi yöntemlerinin başlıcaları da;

- Temel Bileşenler Analizi,
- Diskriminant Analizi,
- Faktör Analizi,
- Kohonen Ağları,
- Bulanık Mantığa Dayalı Yöntemler,
- Genetik Algoritmalar,
- Bayesci Ağlar,
- Pürüzlü (Rough) Küme Teorisine Dayalı yöntemler,

olarak sıralanabilir.

Özet olarak, bilgi keşfine yarayan her yöntem veri madenciliği ve makine öğrenmesi yöntemi olarak kullanılabilmektedir.” [2]

Yukarıdaki bilgilerin ışığında bu çalışmamda kullanmış olduğum yöntem, problemin başlığı temelde sınıflandırma olarak adlandırılabilir. Sınıflandırma (Classification) aslında bir makine öğrenme algoritmalarının bir başlık adıdır. Bu gruba dahil olan algoritmaların temeldeki işlevi; etiket ve kategorik veriler üzerinden belirli parametreler dikkate alınarak eğitilip daha sonra makineden bir sonraki etiketin tahminini yapması beklenmektedir. Bu kategorik veriler veya etiket olarak adlandırdığımız parametreler benim çalışmamda, gelen mesajın spam veya ham olma durumudur. Aşağıda hem yaygın kullanıma sahip, hem de projemde kullanmış olduğum başlıca veri madenciliği ve makine öğrenmesi yöntemlerinin kısa tanımları verilmiştir.

3.1.4.1.Regresyon Analizi

Yaygın olarak kullanılan bir modelleme tekniğidir. Doğrusal, doğrusal olmayan ve lojistik modelleme alternatifleri imkanı vardır.

Regresyon problemi, çıktı değişkeninin “para birimi” veya “ağırlık” gibi gerçek bir değer olduğu durumudur. Yani bir değer (genellikle sürekli olan değerler) arıyorsak, buna regresyon denir. Mesela predictors olarak adlandırılan bir dizi özellik (kilometre, yaş, marka vb.) verildiğinde, bir otomobilin fiyatı gibi bir sayısal değer tahmin etmektir. İşte bu tür bir göreve regresyon denir. Bir sistemi eğitmek için, hem özellikler dizisi (predictors) hem de etiketler (label) dahil olmak üzere birçok araba örneği vermemiz gerekmektedir.

Bazı regresyon algoritmalarının sınıflandırma için de kullanılabileceğini ve sınıflandırma algoritmalarının da regresyon için kullanılabileceğini unutmamalıyız. Örneğin, Lojistik Regresyon, belirli bir sınıfa ait olma olasılığına karşılık gelen bir değer olabileceğinden, sınıflandırma için yaygın olarak kullanılır.

Çalışmamda regresyon analizinden lojistik regresyonu kullanıp, 0.9829596412556054 (%98.3) değerinde bir başarı oranı yakaladım. Bu değerin hesaplanmasında, Lojistik Regresyon' un içerisine aldığı önemli bir parametre olan *solver* önemli bir parametredir. Solver parametresi aslında doğrudan bizim veri

[2] Ali Serhan Koyuncugil, Nermin Özgülbaş, Veri Madenciliği: Tıp ve Sağlık Hizmetlerinde Kullanımı ve Uygulamaları, Bilişim Teknolojileri Dergisi, Cilt: 2, Sayı:2, Mayıs 2009, s. 26-27

setimizle alakalıdır. Solver, algoritmanın veri setiyle olan optimizasyonunu belirler. Ve altında bir matematik; türevler, fonksiyonlar, paraboller ile hesaplanan bir yaklaşım yatmadır. Parametrenin aldığı değerlerden bahsedecek olursak; elimizdeki veri seti bu çalışmada kullandığım veri seti gibi fazla büyük değilse “liblinear” iyi bir seçimdir. Çoklu sınıflandırma problemlerinde ise “newton-cg”, “sag”, “saga” ve “lbfgs” parametre değerleri, liblinear' a göre daha başarılıdır.

3.1.4.2.En Yakın K Komşuluk (K-Nearest Neighbors)

K-en yakın komşuluk (KNN) algoritması, uygulaması kolay gözetimli öğrenme algoritmalarındandır. Hem sınıflandırma hem de regresyon problemlerinin çözümünde kullanılır. Algoritma, sınıfları belli olan bir örnek kümesindeki verilerden yararlanılarak kullanılmaktadır. Örnek veri setine katılacak olan yeni verinin, mevcut verilere göre uzaklığı hesaplanıp, k sayıda yakın komşuluğuna bakılır. KNN; eski, basit ve gürültülü eğitim verilerine karşı dirençli olması sebebiyle en popüler makine öğrenme algoritmalarından biridir. Fakat bunun yanında dezavantajı da mevcuttur. Örneğin, uzaklık hesabı yaparken bütün durumları sakladığından, büyük veriler için kullanıldığında çok sayıda bellek alanına gereksinim duymaktadır.

KNN algoritmasında, içerisine aldığı "n_neighbors" parametresi bulunmaktadır. Bu parametre; yeni gelen verinin eski verilere ya da veriye olan uzaklığı olarak anlamlandırılır. Ve bu değer bize yeni gelen verinin eski verilerden kaç tanesine komşu, kaç tanesine bakılarak bir etikete sahip olacağı anlamına gelir. Bu değere 1 vermek ile 5 vermek çok farklı sonuçlar görmemize yol açacaktır. Burada da yine değer denemesi yapılarak en başarılı sonucu veren değer tercih edilmeli ve model bunun üzerine kurulmalıdır.

Bu çalışma için kullandığım KNN algoritması, oluşturduğum model üzerinde 0.968609865470852 (%96.8) oranında başarı sağladı. Bu değer yukarıda bahsettiğim bütün parametre ve yöntemlerle değişken olup, bu model için en iyi sonuçtur.

3.1.4.3.Destek Vektör Makineleri (Support Vector Machines (SVMs))

Destek Vektör Makineleri istatiksel öğrenme teorisine dayalı bir gözetimli öğrenme algoritmasıdır. Temel olarak iki sınıfa ait verileri birbirinden en uygun şekilde ayırmak için kullanılır. Bunun için karar sınırları yada diğer bir ifadeyle hiper düzlemler belirlenir. DVM'ler günümüzde yüz tanıma sistemlerinden, ses analizine kadar birçok sınıflandırma probleminde kullanılmaktadırlar.

Svm's algoritmasının içerisine aldığı, çekirdek fonksiyonu olarak tanımlayabileceğimiz kernel parametresi bulunmaktadır. İçerisine 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' olmak üzere 5 farklı özellikte değer alabilmektedir. Bu parametre değerlerinin hepsinin kendi içerisinde matematiksel hesapları ve incelikleri bulunmaktadır. Ama kullandığımız rbf' yi anlatmak gerekirse; bir SVM modelini Radyal Temel Fonksiyonu (RBF) çekirdeği ile eğitirken, iki parametre dikkate alınmalıdır: C ve gamma. C Tüm SVM çekirdeklerinde ortak olan parametre, eğitim örneklerinin yanlış sınıflandırmasını karar yüzeyinin basitliğine karşı değiştirir. Düşük C, karar yüzeyini pürüzsüzleştirir, yüksek C ise tüm eğitim örneklerini doğru şekilde sınıflandırmayı amaçlar. Gamma tek bir eğitim örneğinin model üzerinde ne kadar etkisi olduğunu tanımlar. Gamma ne kadar büyükse, diğer örneklerin etkilenmesi o kadar fazla (yakın) olmalıdır. Doğru C ve gamma seçimi SVM' nin performansı için çok önemlidir.

SVMs algoritması, modelim üzerinde 0.9856502242152466 (%98.5) başarı oranı yakaladı. Bu değerin yukarıda anlatılanlar neticesinde değişkenlik göstermesi olası olup, bu model için en iyi sonuçtur.

3.1.4.4.Karar Ağaçları (Decision Trees)

Ağaç tabanlı öğrenme algoritmaları, en çok kullanılan gözetimli öğrenme algoritmalarındandır. Genel itibariyle ele alınan bütün problemlerin (sınıflandırma ve regresyon) çözümüne uyarlanabilirler. Fakat bu model kayıp değerleri desteklememektedir, bu değerleri önce düzenlememiz, duruma göre ortalama ya da en çok tekrar eden değer gibi değer atamaları yapmamız gerekmektedir. Karar ağacı algoritması, veri madenciliği sınıflandırma algoritmalarından biridir. Önceden tanımlanmış bir hedef değişkene sahiplerdir. Yapıları itibariyle en tepeden en aşağı inen bir strateji sunmaktadırlar. Bir karar ağacı, çok sayıda kayıt içeren bir veri kümesini, bir

dizi karar kuralları uygulayarak daha küçük kümelere bölmek için kullanılan bir yapıdır. Yani basit karar verme adımları uygulanarak, büyük miktarlardaki kayıtları, çok küçük kayıt gruplarına bölerek kullanılan bir yapıdır. Buradaki önemli husus, karar ağaçlarının en büyük problemi aşırı öğrenme, veriyi ezberlemedir (overfitting). İşte tam bu noktada devreye “Random Forests” girmektedir.

Karar Ağaçları algoritması, bu çalışmadaki model üzerinde 0.979372197309417 (%97.9) başarı oranı göstermiş, yine bu oran bu model için en optimize olup yukarıda sayılan niceliklere göre değişebilmektedir.

3.1.4.5.Rastgele Ormanlar (Random Forests)

Random Forests modeli bu problemi çözmek için hem veri setinden hem de öznitelik setinden rastgele olarak 10'larca 100'lerce farklı alt setler seçiyor ve bunları eğitiyor. Bu yöntemle 100'lerce karar ağacı oluşturuluyor ve her bir karar ağacı bireysel olarak tahminde bulunuyor. Günün sonunda problemimiz regresyonsa karar ağaçlarının tahminlerinin ortalamasını, problemimiz sınıflandırmaysa tahminler arasında en çok oy alan etiketi seçiyoruz. Şimdi bütün bunları basit bir örnekle açıklayacak olursak: Örneğin bu akşam güzel bir film izlemek istiyorsunuz ve kafanız karışık. Bir arkadaşınızı ararsanız ve o size tercih ettiğiniz film türü, süre, yıl, oyuncu-yönetmen, Hollywood-alternatif vs. soru setinden çeşitli sorularla daha önce izlediğiniz filmlere (training set) göre bir tahminde bulunursa bu karar ağacı olur. Eğer 20 arkadaşınız bu soru setinden farklı sorular seçip verdiğiniz cevaplara göre tavsiyede bulunursa ve siz en çok tavsiye edilen filmi seçerseniz bu Random Forests modelimiz olur. Random Forests modelinde farklı veri setleri üzerinde eğitim gerçekleştiği için varyans, diğer bir deyişle karar ağaçlarının en büyük problemlerinden olan overfitting azalır. Random Forests modelinin diğer bir özelliği bize özniteliklerin ne kadar önemli olduğunu vermesi. (Bir özniteliğin önemli olması demek o özniteliğin bağımlı değişkendeki varyansın açıklanmasına ne kadar katkı yaptığıyla alakalı bir durumdur.) Random Forests algoritmasına x sayıda öznitelik verip en faydalı y tanesini seçmesini isteyebiliriz ve istersek bu bilgiyi istediğimiz başka bir modelde kullanabiliriz.

Random Forests algoritmasının içerisinde bulunan ve bize oluşturulmak istenen ağaç sayısını belirten n_estimators parametresi sayesinde oluşturmak istediğimiz ağaç sayılarını belirleyip algoritmayı buna göre şekillendirebiliyoruz. Buradaki değeri ne

kadar arttırırsak o kadar iyi performans alabiliriz. Tabi bu veri setinin büyüklüğü ile de alakalı bir durumdur. Sonuçta bu ağaç sayılarını arttırmanın da bir maliyeti vardır ve kodumuzu oldukça yavaşlatabilir. Bu yüzden veri setiyle optimize olacak şekilde ayarlanmalıdır.

Random Forests algoritması, bu çalışmadaki model üzerinde 0.9865470852017937 (%98.6) başarı oranı göstermiş, yine bu oran bu model için en optimize olup yukarıda sayılan niceliklere göre değişebilmektedir.

3.1.4.6 Naive Bayes

Bu algoritma, koşullu olasılıklara dayanarak hedef sınıftaki belirli bir değerin gerçekleşmesi ihtimalini inceler ve buna dayalı olarak hedef sınıfın değerini tahmin eder. Bu hesaplama Bayes Formülü aracılığı ile gerçekleştirilir. Naive Bayes algoritması sonuç olarak hedef sınıfın hangi değerinin gerçekleşme olasılığının ne olduğunu bildirir. Bir defa sistem eğitildikten sonra oldukça performanslı çalışsa da sistemin ilk kez eğitilmesi veya eğitilmiş bir sistemin güncellenmesi için yeniden eğitilmesi ise zaman ve kaynak kullanımı anlamında maliyetlidir. Çünkü tüm veri kümesini her sınıflandırma işlemi için tekrar tekrar taraması gerekir. Naive Bayes algoritması Metin Madenciliğine (Text Classification ve Multinomial Naive Bayes) ve Çoklu Sınıf Tahminlerine (Multi Class Prediction) elverişlidir.

Naive Bayes algoritması, bu çalışmadaki model üzerinde 0.9856502242152466 (%98.5) başarı oranı göstermiştir.

SONUÇ

Çalışmamızın amacı doğrultusunda 6 farklı algoritma model üzerinde uygulanmış ve gözlemlenmiştir. Gözlemlere göre model üzerindeki algoritmaların başarı sıralamaları şu şekildedir:

1. Random Forests (%98.6)
2. Naive Bayes (%98.5)
3. Support Vector Machines (SVMs) (%98.5)
4. Logistic Regression (%98.3)
5. Decision Trees (%97.9)
6. K-Nearest Neighbors (KNN) (%96.8)

Gözlemler sonucu model üzerinde en başarılı olan algoritma Random Forests (%98.6), model üzerinde başarı oranı en düşük algoritma ise KNN algoritması olmuştur. Bu sonuçlar algoritmaların içerisinde kendilerine has önemli parametreler ve kullandığımız veri seti ile bağlantılıdır. Başka bir problemin başka bir veri seti ile algoritma başarı sıraları değişebileceği gibi farklı veri madenciliği veya metin madenciliği yöntemleri de sonucu etkileyebilir. Bu nedenle aynı veri seti ve aynı problem üzerinde çalışan iki farklı kişinin birbirinden farklı değerli başarı oranlarını aynı algoritmayı kullanarak da bulması olası bir durumdur.

KAYNAKÇA

- G. Piatetsky-Shapiro, W. J. Fawley, “Knowledge Discovery in Databases”, AAAI/MIT Pres, 1991.
- U. Fayyad, G. Piatetsky-Shapiro, P. Symth, P. “From Data Mining to Knowledge Discovery in Databases”, AI Magazine, 17(3), 37- 54, 1996.
- A.S. Koyuncugil, “Bulanık veri madenciliği ve sermaye piyasalarına uygulanması”, Doktora tezi (basılmamış), Ankara Üniversitesi, Fen Bilimleri Enstitüsü, 2006.
- P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, A. Zanasi, “Discovering Data Mining: From Concept To Implemantation”, Prentice Hall PTR, Upper Saddle River, New Jersey, 195, USA. 1997.
- A. Berson, S. Smith, K. Thearling, “Buildind Data Mining Applications for CRM”,. Mcgraw Hill, 510, USA, 1999.
- T. Hastie, R. Tibshirani, J. Friedman, “The Elements Of Statistical Learning; Data Mining, Inference And Prediction”, Springer Series In Statistics, 533, USA, 2001.
- Internet, K. Thearling, www.thearling.com. Erişim Tarihi: 18.06.2005.
- Z. Chen, “Data Mining And Uncertain Reasoning: An Integrated Approach”,. John Wiley & Sons, Inc., 370, Canada. 2001.
- B. Kovalerchuk, E. Vityaev, “Data Mining in Finance: Advances in Relational And Hybrid Methods”. Kluwer Academic Publishers, 308, USA. 2001.
- Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O’Reilly Media Inc, https://www.nltk.org/
- Python ile Makine Öğrenimi, https://medium.com/turkce/python-ile-makine-%C3%B6%C4%9Frenimi-237befcfc5b