**Veri Madenciliği ile Sınıflandırma Proje Raporu**

**Hazırlayan**

Fırat Kaan Bitmez

**Öğrenci Numarası**

23281855

**Dersin Hocası**

Prof.Dr. Erdal Kılıç

**Giriş**

Bu Rapor, Veri madenciliği dersinde verilen Sınıflandırma projesi için hazırlanmıştır. Proje, Türkçe köşe yazarlarının yazılarını sınıflandırmayı amaçlamaktadır. Beş farklı köşe yazarının en az 20 köşe yazısı kullanılarak bir model oluşturulmuş ve bu model, yeni köşe yazılarının hangi yazarlara ait olduğunu tahmin etmek için kullanılmıştır.

**Amacımız**

Bu projenin temel amacı, metin sınıflandırma algoritmalarını kullanarak Türkçe köşe yazarlarının yazılarını sınıflandırmaktır. Bunun için, veri madenciliği teknikleri ve doğal dil işleme yöntemleri kullanılmıştır.

**Kullanılan Kütüphaneler, Araçlar ve Modüller**

Proje **Python** programa dili ile kodlanmıştır.

Projede kullanılan Kütüphane ve Modüller aşağıda listelenmiştir. Parantez içerisinde Yükleme komutları verilmiştir. Terminal üzerinden bu kütüphaneleri direk komutu kopya yapıştır ile yükleyebilirsiniz.

**collections**: Veri yapıları için kullanılmıştır.

**os**: Dosya işlemleri için kullanılmıştır. (pip install os)

**re**: Regular expressions için kullanılmıştır. (pip install regex)

**numpy**: Sayısal işlemler için kullanılmıştır. (pip install numpy)

**snowballstemmer**: Türkçe kök bulma işlemleri için kullanılmıştır.( pip install snowballstemmer)

**sklearn**: Model oluşturma, metrikler ve sınıflandırma algoritmaları için kullanılmıştır. (pip install scikit-learn)

**seaborn**: Görselleştirme için kullanılmıştır.

**matplotlib.pyplot:** Görselleştirme için kullanılmıştır. (pip install matplotlib)

**Kullanılan Algoritma ve Araçlar**

Projede kullanılan ana algoritma, sınıflandırma için **Destek Vektör Makineleri (Support Vector Machine)** algoritmasıdır**. SVM,** metin sınıflandırma gibi çoklu sınıflı problemlerde etkili bir şekilde kullanılabilir**.** Model oluşturulurken, metin verileri TF-IDF (Term Frequency-Inverse Document Frequency) vektörleme yöntemiyle sayısal vektörlere dönüştürülmüştür. Bu vektörler, SVM algoritması tarafından sınıflandırma için kullanılmıştır.

**Tasarım Adımları**

1. **Veri Toplama ve Düzenleme**

Projedeki Veri Kümesi, beş farklı köşe yazarının 24’er Köşe yazısı alınarak **Toplamda 120 Köşe yazısından** oluşan bir Data oluşturulmuştur. Her bir yazarın Eğitimi için 20, Testi için 4’er köşe yazısı ayrılmıştır. Bu Projede **Eğitim için 100 Köşe yazısı test için ise 20 köşe yazısı** kullanılmıştır.

**Eğitim** için ayrılan Köşe yazıları Data klasörü içinde yazar1, yazar2, yazar3, yazar4, yazar5 klasörleri içerisinde **20’şer** tane olmak üzere **100** köşe yazısı toplanmış ve düzenlenmiştir.

**Test** için ayrılan köşe yazıları ise **testyazar** klasörü içerinde toplanmıştır.

Her yazarın köşe yazıları doc(document) kısaltmasıyla adlandırılmış ve 1’den 20 ye kadar numaralandırılarak Veri Setimiz kategorize ve düzenli bir hale getirilmiştir.

**metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Veri Setimizde Eğitim ve Test için kullanacağımız yazarların Kimlikleri şu şekildedir.

Yazar1: **Uğur Dündar**

Yazar2: **Soner Yalçın**

Yazar3: **Murat Muratoğlu**

Yazar4: **Necati Doğru**

Yazar5: **Rahmi Turan**

Test için ayrılan 20 köşe yazısı ise yine aynı şekilde 1’den 20’ye kadar olacak şekilde sıralanmış ve **testyazar** klasörü içerisinde toplanmıştır. Buradaki test isimli text dosyalarını 4’er 4’er olacak şekilde yazar1, yazar2, yazar3, yazar4, yazar5 diye adlandırılmış yazarlara aittir.

Yani testyazar klasöründeki test dosyalarının:

* **1-4** Arası köşe yazılarını yazarı **Yazar1** (Uğur Dündar),
* **5-8** Arası köşe yazılarını yazarı **Yazar2** (Soner Yalçın),
* **9-12** Arası köşe yazılarını yazarı **Yazar3** (Murat Muratoğlu),
* **13-16** Arası köşe yazılarını yazarı **Yazar4** (Necati Doğru),
* **17-20** Arası köşe yazılarını yazarı **Yazar5** (Rahmin Turandır.

**metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Veri Setimizi oluşturduktan sonra **Önişleme** ve **Temizleme** adımları için bir **Stopword** Türkçe Kelimelistesi oluşturdum ve Proje içerisine dahil ettim. Bu Kelime Listesi **Zemberek** Doğal Dil İşleme Kütüphanesi içerisinden alınmıştır.

**stopword.txt** diye adlandırdığım ve proje içerisine dahil etmek için projede şöyle bir kodlama yapıldı.

# Stop word dosyasını okuyarak, metinlerdeki gereksiz kelimeleri filtrelemek için bir stop word listesi

with open(stopwords\_path, "r", encoding="utf-8") as stopwords\_file:

    stop\_words = stopwords\_file.read().splitlines()

1. **Veri Temizleme ve Önişleme**

Veri temizleme ve ön işleme adımı, metin verilerinin analiz ve modelleme sürecine hazırlanmasını sağlayan önemli bir aşamadır. Bu adımda, veri setindeki metinlerin anlamlı bilgilerini çıkarmak ve gereksiz bilgilerden arındırmak için çeşitli işlemler uygulanır. İşte bu adımların detaylıca açıklaması:

* **Özel Karakterlerin Kaldırılması**

Metin verilerinde genellikle noktalama işaretleri, parantezler, tireler gibi özel karakterler bulunur. Bu karakterler metin analizinde genellikle gereksizdir ve model performansını olumsuz etkileyebilir. Bu nedenle, özel karakterler genellikle kaldırılır.

 cleaned\_text = re.sub(r'\W', ' ', text)

* **Sayıların Kaldırılması**

Metin verilerinde sayılar bulunabilir, ancak metin sınıflandırma için genellikle anlamsızdır. Bu nedenle, metin verilerinden sayılar genellikle kaldırılır.

 cleaned\_text = re.sub(r'\d+', ' ', cleaned\_text)

* **Küçük Harfe Dönüştürme**

Metin verilerinin tümünü küçük harflere dönüştürmek, büyük ve küçük harf farklılıklarını ortadan kaldırarak modele daha tutarlı bir şekilde erişmesini sağlar. Örneğin, "Kitap" ve "kitap" kelimeleri aynı kelime olarak kabul edilir.

cleaned\_text = cleaned\_text.lower()

* **Stop Words'lerin Kaldırılması**

Stop words, metinlerde sıklıkla görülen ancak genellikle anlam taşımayan kelimelerdir (örneğin, "ve", "ama", "veya" gibi). Bu kelimeler genellikle modelin performansını düşürür ve bu nedenle genellikle kaldırılır.

cleaned\_text = ' '.join([word for word in cleaned\_text.split() if word not in stop\_words])

* **Türkçe Kök Bulma İşlemi**

Türkçe kök bulma işlemi, kelimelerin köklerini bulmayı amaçlar. Örneğin, "geliyorum", "geliyorsun", "geliyorlar" kelimelerinin kökü "gel"dir. Bu işlem, modelin daha genelleştirilmiş ve anlamlı özelliklerle çalışmasını sağlar.

  stemmer = TurkishStemmer()

                cleaned\_text = ' '.join([stemmer.stemWord(word) for word in cleaned\_text.split()])

Bu adımların uygulanması, metin verilerinin temizlenmesini ve modele hazır hale getirilmesini sağlar. Bu sayede, modelin daha iyi performans göstermesi ve doğru sonuçlar üretmesi sağlanır.

 # Eğitim verisine temizlenmiş metni ekleyin

                training\_data[author\_name].append(cleaned\_text)

1. **TF-IDF Vektörleme**

TF-IDF (Term Frequency-Inverse Document Frequency), metin verilerini sayısal vektörlere dönüştürmek için yaygın olarak kullanılan bir tekniktir. Bu teknik, metin verilerindeki her bir kelimenin önem derecesini belirlemek için kullanılır.

# Model oluşturma

#Pipeline, ardışık işlemleri sıralı olarak gerçekleştirmenizi sağlar. Bu durumda, TF-IDF vektörleme işlemi ve SVC (Support Vector Classifier) sınıflandırma algoritması sırayla uygulanır.

model = Pipeline([

    ('tfidf', TfidfVectorizer()),

    ('classifier', SVC())

])

Burada, Pipeline kullanılarak TfidfVectorizer() sınıfı TF-IDF vektörleme işlemi için kullanılmıştır. Bu şekilde, tfidf adı altında vektörleme işlemi, diğer sınıflandırma adımı olan classifier adı altında ise SVC (Destek Vektör Makineleri) sınıflandırma algoritması ile bir boru hattı (pipeline) oluşturulmuştur.

İlk olarak, TF (Term Frequency) ve IDF (Inverse Document Frequency) kavramlarını ayrı ayrı açıklayarak başlayalım:

* **Term Frequency (TF - Kelime Sıklığı)**

Bir belgedeki bir kelimenin ne kadar sıklıkta geçtiğini gösteren bir metrik. Genellikle, bir belgedeki bir kelimenin sıklığı, o kelimenin belgedeki toplam kelime sayısına oranı olarak hesaplanır. Ancak bazen kelime frekansı doğrudan kelimenin belgedeki toplam sayısını da temsil edebilir. Örneğin, "kitap" kelimesinin bir belgedeki sıklığı, o belgede "kitap" kelimesinin geçtiği toplam sayıdır.

* **Inverse Document Frequency (IDF - Ters Belge Frekansı)**

Bir kelimenin belgedeki nadirliğini ölçen bir metrik. IDF, bir kelimenin ne kadar nadir olduğunu belirler. Nadir kelimeler, belgedeki genel içeriği daha iyi temsil edebilir ve bu nedenle daha yüksek bir öneme sahip olabilirler. IDF, bir kelimenin belgedeki tüm belgelerde ne kadar nadir olduğunu hesaplar ve bu nadirlik derecesine göre bir ağırlık verir. Nadir kelimelerin IDF değerleri daha yüksektir.

TF-IDF vektörleme işlemi, TF ve IDF değerlerinin çarpımıyla elde edilir ve bu işlem her bir kelimenin her bir belge için bir özellik vektörü oluşturur. TF-IDF vektörleme adımları şu şekilde gerçekleşir:

* **TF-IDF Vektörleme**

TF ve IDF değerleri, her bir kelimenin her bir belgedeki önemini belirlemek için çarpılır. Böylece, her bir belge için bir TF-IDF vektörü oluşturulur.

Bu işlem sonucunda, her bir belge TF-IDF vektörlerinin oluşturulmasıyla sayısal bir temsile dönüştürülür. Bu sayısal temsil, metin verilerinin modellemeye uygun hale getirilmesini sağlar ve metin sınıflandırma gibi görevler için kullanılabilir. TF-IDF vektörleme, metin verilerindeki kelime önemini vurgulayarak modelin daha iyi performans göstermesini sağlar.

1. **Destek Vektör Makineleri (SVM) Algoritması ile SVC (Support Vector Classifier) Sınıflandırma**

Support Vector Machine (Destek Vektör Makineleri diğer adıyla DVM ya da SVM) sınıflandırma, regresyon ve aykırı değerleri bulmak için kullanılan denetimli (superwised) bir öğrenme tekniğidir. SVM algoritması classification kavramı adı altında gelişen ve diğer classification türlerinden farklı olan bir algoritmadır.  SVM algoritması bir düzlem üzerine yerleştirilmiş 2 veya daha fazla nokta kümelerini ayırmak için doğrular çizer. 2 veri kümesi düşünüldüğünde bu doğrunun, iki kümenin noktaları için de maksimum uzaklıkta olmasını amaçlar. Karmaşık ama küçük ve orta ölçekteki veri setleri için uygundur.

çizgi, diyagram, ekran görüntüsü, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görüldüğü üzere iki veri kümesi grafik üzerinde görülmekte ve 3 doğrusal çizgi. Buradaki öncelikli amacımız veri kümelerinin ayıran bir doğru çizmek. Görüldüğü üzere L\_1 doğrusu bu iki kümeyi ayırmakta en başarısız olanı. Diğer iki doğruya bakacak olursak L\_2 mavi renkli noktaların oluşturduğu kümeye daha yakınken kırmızı veri kümesine daha uzak kalır. Buradaki dengesizlik yeni tahmin edilmesi istenen verinin daha hatalı bir tahminle sonuçlanmasına neden olacaktır. Bundan dolayı algoritma 2 veri kümesindeki birbirlerine en yakın olan verilerin (kırmızı ve mavi verilerin) arasındaki en fazla aralığı yakaladığı doğruyu çizmeyi tercih edecektir. Bu arada sınıflar arası çizilebilecek sonsuz adet doğru vardır ve bu çizilebilecek doğrulara karar doğruları adı verilir.

metin, diyagram, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu iki sınıfı en iyi ayıran doğrunun (L\_3) bölgeler arasında kalan alana margin denir. Margin ne kadar geniş olursa sınıflar okadar iyi ayrıştırılır. Bazı durumlarda veirler margin bölgesine girebilir. Bu duruma ‘Soft Margin’ denir. Hard Margin ise veri doğrusal olarak ayrılabiliyorsa çalışır.

metin, diyagram, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada görüldüğü gibi her iki veri kümesine destek vektörleri çizilmiştir. Bu destek vektörleri referans alınarak SVM çizilebilecek en iyi doğruyu çizer (w.x+b=0). SVM lerde sınıflar +1 veya -1 olarak etiketlenir. Bundan dolayı karar doğrusu (hiper düzlem) üstünde kalan doğruya wx+b=1 altında kalan doğruya ise wx+b=-1 olarak yazılır.

**Kısaca Özetlemek gerekirse SVM**, diğer yöntemlere göre iki temel avantaja sahiptir. Yüksek boyutlu uzaylarda işlem yapabilme yeteneğine sahiptir ve bu, genellikle birçok özelliğin kullanılması durumunda yararlıdır. Ayrıca, aşırı uymaya karşı dayanıklıdır, yani genelleme yeteneği daha iyidir.

**Bu kodlamada Destek Vektör Makineleri (SVC) sınıflandırması için aşağıdaki kodlar bulunmaktadır**:

* **SVM Algoritmasıyla SVC’nin oluşturulması**

model = Pipeline([

    ('tfidf', TfidfVectorizer()),

    ('classifier', SVC())

])

* **En iyi parametrelerin bulunması için GridSearchCV kullanımı ve en iyi modelin seçilmesi**

# GridSearchCV ile en iyi parametreleri bulma

#GridSearchCV, bir model için en iyi hiperparametreleri bulmanızı sağlayan bir cross-validation aracıdır. Bu durumda, SVC için C, gamma ve kernel parametreleri için farklı değerler denenecek ve en iyi parametreler bulunacaktır.

param\_grid = {

    'classifier\_\_C': [0.1, 1, 10, 100],

    'classifier\_\_gamma': [1, 0.1, 0.01, 0.001],

    'classifier\_\_kernel': ['rbf', 'linear', 'poly', 'sigmoid']

}

grid\_search = GridSearchCV(model, param\_grid, cv=5)

grid\_search.fit(X, y)

print("GridSearchCV ile en iyi parametrelerin bulunması:")

print("------------------------------------------------")

print("En iyi parametreler:", grid\_search.best\_params\_)

print("En iyi skor:", grid\_search.best\_score\_)

print("\n")

best\_model = grid\_search.best\_estimator\_

**5.Modelin Eğitilmesi ve Değerlendirilmesi**

Modelin eğitilmesi ve değerlendirilmesi, makine öğrenimi projelerinde kritik bir aşamadır. Bu aşama, modelin performansının anlaşılmasına ve geliştirilmesine olanak sağlar. İşte bu adımların detaylı bir açıklaması:

* **Eğitim Aşaması**

Model eğitimi, öncelikle belirlenen algoritmaya ve parametrelere göre gerçekleştirilir. Eğitim veri seti kullanılarak model oluşturulur ve bu model, veri setindeki örüntüleri öğrenir.

Destek Vektör Makineleri (SVM) gibi algoritmalar için, eğitim aşamasında hiperdüzlem belirlenir ve modelin parametreleri ayarlanır.

#En iyi model seçilir ve eğitim verisi ile eğitilir. Ayrıca, doğrulama verisi için train\_test\_split kullanılarak veri seti bölünür.

# Eğitim ve doğrulama verisi için modeli eğitin

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

best\_model.fit(X\_train, y\_train)

* **Doğrulama Aşaması**

Model oluşturulduktan sonra, genellikle ayrılmış olan doğrulama veri seti kullanılarak modelin performansı değerlendirilir. Bu aşamada, modelin ne kadar iyi genelleştirildiği ve yeni veri noktaları üzerinde ne kadar başarılı olduğu ölçülür.

Doğrulama veri seti üzerinde modelin performansını değerlendirmek için çeşitli metrikler kullanılabilir. Bunlar arasında doğruluk, hassasiyet, geri çağırma ve F1-score gibi değerler bulunur.

#Modelin performansını değerlendirmek için confusion matrix hesaplanır.

y\_pred = best\_model.predict(X\_val)

conf\_matrix = confusion\_matrix(y\_val, y\_pred)

#Confusion matrix, görselleştirilerek daha anlaşılır hale getirilir.

plt.figure(figsize=(10, 8))

sns.heatmap(conf\_matrix, annot=True, cmap="Blues", fmt="d", xticklabels=best\_model.classes\_, yticklabels=best\_model.classes\_)

plt.xlabel('Tahmin Edilen')

plt.ylabel('Gerçek Değer')

plt.title('Confusion Matrix')

plt.show()

* **Test Aşaması**

Modelin eğitim ve doğrulama aşamalarını tamamladıktan sonra, genellikle ayrılmış olan test veri seti kullanılarak modelin performansı kesin olarak değerlendirilir.

Test veri seti, genellikle modelin daha önce görmediği verilerden oluşur ve modelin gerçek dünya performansını yansıtır.

Modelin test veri seti üzerindeki performansı, eğitim ve doğrulama aşamalarında elde edilen sonuçlarla karşılaştırılarak değerlendirilir.

# Test verisi üzerinde tahminler

print("Tahminler:")

print("----------------------------")

predictions = []

for i, file\_name in enumerate(sorted(os.listdir(test\_data\_folder), key=lambda x: int(re.findall(r'\d+', x)[0]))):

    file\_path = os.path.join(test\_data\_folder, file\_name)

    with open(file\_path, "r", encoding="utf-8") as file:

        text = file.read()

        # Test metnini temizleme ve kök bulma işlemleri

        cleaned\_text = re.sub(r'\W', ' ', text)

        cleaned\_text = re.sub(r'\d+', ' ', cleaned\_text)

        cleaned\_text = cleaned\_text.lower()

        cleaned\_text = ' '.join([word for word in cleaned\_text.split() if word not in stop\_words])

        stemmer = TurkishStemmer()

        cleaned\_text = ' '.join([stemmer.stemWord(word) for word in cleaned\_text.split()])

        # Model ile tahmin yapma

        prediction = best\_model.predict([cleaned\_text])[0]

        predictions.append((file\_name, prediction))

# Tahminleri detaylı olarak yazdırma

for file\_name, prediction in predictions:

    print(f"{file\_name}: Tahmin Edilen Yazar: {prediction}")

* **Performans Değerlendirme**

Modelin performansı, doğrulama ve test aşamalarında elde edilen sonuçlar kullanılarak değerlendirilir.

Doğruluk, hassasiyet, geri çağırma, F1-score gibi metriklerin yanı sıra ROC eğrisi, confusion matrix gibi görsel araçlar da kullanılarak modelin performansı analiz edilir.

Bu analiz sonucunda, modelin hangi sınıflarda daha iyi veya daha kötü performans gösterdiği belirlenir ve gerektiğinde modelin iyileştirilmesi için önlemler alınır.

Modelin eğitilmesi ve değerlendirilmesi aşamaları, makine öğrenimi projelerinde modelin başarısını belirlemek için kritik öneme sahiptir. Bu aşamaların titizlikle yürütülmesi, projenin başarısını ve modelin gerçek dünya performansını doğrudan etkiler.

ekran görüntüsü, metin, diyagram, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Sonuç**

Eğitim verileri temizlenmiş ve kök bulma işlemleri uygulanmıştır. Model, GridSearchCV ile en iyi parametrelerin bulunmasıyla oluşturulmuştur. Son olarak, model test verisi üzerinde değerlendirilmiştir ve başarı metrikleri kullanılarak performansı değerlendirilmiştir.

Projede elde edilen sonuçlar oldukça başarılıdır. Model, test verisi üzerinde %80 civarında bir doğruluk elde etmiştir. Confusion matrisi ve sınıflandırma raporu ile modelin performansı detaylı bir şekilde incelenmiştir. Proje, Türkçe köşe yazarlarının metinlerini sınıflandırmak için etkili bir çözüm sunmuştur.

**Kaynaklar**

<https://github.com/firatkaanbitmez/veri-madenciligi>

<https://mfatihto.medium.com/support-vector-machine-algoritması-makine-öğrenmesi-8020176898d8>