Özet:

Bu projede, farklı makale türlerini sınıflandırmak için bir veri madenciliği modeli geliştirilmiştir. Proje sürecinde, beş farklı türde (bilişim, tıp, tarih, ekonomi ve spor) toplamda minimum 100 makale içeren bir eğitim kümesi oluşturulmuştur. Zemberek kütüphanesi kullanılarak metin ön işleme yapılmış ve ardından eğitim verileri ve test verileri oluşturulmuştur. Proje sonunda, geliştirilen model yeni bir makale verildiğinde makalenin hangi türe ait olduğunu başarıyla tespit edebilmektedir.

Veri Kümesi Oluşturma

Projede, her biri farklı bir türde olan toplam 126 makale kullanılmıştır. Bu makaleler, 21 bilim, 21 teknoloji, 21 sağlık, 21 tarih, 21 ekonomi ve 21 spor makalesi olmak üzere altı farklı türde oluşturulmuştur. Veri kümesi oluşturma sürecinde, her makale türünden eşit sayıda örnek kullanılarak dengeli bir veri kümesi elde edilmiştir.

Ön İşleme

Ön işleme sürecinde, Zemberek kütüphanesi kullanılarak metinlerdeki yazım hataları düzeltilmiş, cümleler ayrılmış ve kök kelimeler çıkarılmıştır. Daha sonra, Türkçe durdurma kelimeleri ve işlenmemesi gereken karakterler kaldırılarak temizlenmiş metinler elde edilmiştir.

Model Oluşturma

Projede, Naive Bayes sınıflandırıcısı kullanılarak makale türü sınıflandırma modeli geliştirilmiştir. Geliştirilen model, Tfidf vektörleştirici ile dönüştürülen metinler üzerinde eğitilmiştir. Modelin başarısı, doğruluk skoru ile ölçülmüştür.

Sonuçlar ve Değerlendirme

Geliştirilen model ile yapılan tahminlerde, kullanıcının girdiği makale türünü başarıyla tespit edebilmektedir. Doğruluk skoru, modelin performansını gösteren önemli bir ölçüttür ve bu projede elde edilen doğruluk skoru %85 civarındadır.

Sonuç olarak, bu proje kapsamında geliştirilen makale türü sınıflandırma modeli, altı farklı türdeki makaleleri başarıyla sınıflandırabilmektedir. Model, Zemberek kütüphanesinin metin ön işleme ve özellik çıkarımı özellikleri ile güçlendirilmiştir. Modelin başarısı, doğruluk skoru ile ölçülmüş ve yüksek doğruluk skoru elde edilmiştir. İlerleyen çalışmalarda, modelin performansını daha da artırmak adına daha fazla veri kullanarak modelin eğitimi gerçekleştirilebilir ve farklı sınıflandırıcılarla karşılaştırma yapılabilir.

Projenin çalıştırılması ve kullanımı

Proje klasöründe main.py dosyası projemizin başlangıç dosyasıdır. veriSeti.py ise kaynak olarak kullanılan makalelerin yer aldığı veri setimizdir.

Projemizi çalıştırdığınızda sizden sınıflandırma yapılmasını istediğiniz makaleyi girmenizi isteyecektir. Eğer herhangi bir giriş yapmadan direk enter tuşuna basarsanız main.py dosyası içinde “**girdiMetinVarsayilan**” isimli değişkene varsayılan olarak atanmış olan makaleyi sınıflandıracaktır. Aşağıdaki görsel proje ilk çalıştırıldığındaki ekran görüntüsüdür.

```
koksaliyigun@MacBook-Pro-2 veriMadenciligiOdev % /usr/bin/python3 /Users/koksaliyigun/Desktop/veriMadenciligiOde
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/koksaliyigun/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
I|17:47:33.427|Binary lexicon read from resource in 13 ms.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.protobuf.UnsafeUtil (file:/Users/koksaliyigun/Desktop/veriMaden
WARNING: Please consider reporting this to the maintainers of com.google.protobuf.UnsafeUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
I|17:47:33.923|Root lexicon created in 491 ms.
I|17:47:33.925|Dictionary generated in 522 ms
I|17:47:34.305|Initialized in 903 ms.
I|17:47:34.553|Initialized in 248 ms.
Lütfen test etmek istediğiniz makaleyi girin:(Eğer boş enter'a basarsanız varsayılan giriş metni işlenir)
```

Eğer boş enter'a basmaz bir makale girişi yaparsanız girdiğiniz makalenin sınıflandırma işlemi gerçekleşecektir.

```
Lütfen test etmek istediğiniz makaleyi girin:(Eğer boş enter'a basarsanız varsayılan giriş metni işlenir)

Sağlık, insan yaşamının en temel ve değerli unsurlarından biridir. Sağlıklı olmak, kişinin fiziksel, zihinsel
ür.Sağlığı etkileyen birçok faktör vardır. Beslenme, egzersiz, uyku, stres yönetimi, hijyen gibi faktörler, sa
zerinde önemli bir etkiye sahiptir.Sağlıkla ilgili birçok konu vardır. Bunlar arasında hastalıkların teşhisi,
ağlığı, erkek sağlığı, yaşlılık sağlığı ve daha birçok konu yer alır.Sağlık konusunda yapılan araştırmaların s




SONUÇ : SAĞLIK MAKALESİ
DOĞRULUK SKORU:  76.31578947368422
```

Kodlar ve Açıklamaları:

1- Kütüphaneler ve gerekli sınıfların import edilmesi:

```
import jpype as jp, nltk, pandas as pd, numpy as np, dataset
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, svm
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
nltk.download('stopwords')
```

- jpype: Java'daki sınıfları ve metotları Python'da kullanmamıza olanak tanır. Burada Zemberek kütüphanesini kullanabilmek için eklenmiştir.
- nltk: Doğal dil işleme (NLP) işlemleri için kullanılır.
- pandas: Veri analizi ve manipülasyonu için kullanılır.
- numpy: Bilimsel hesaplamalar için kullanılır.
- dataset: Veri kümesini içeren modüldür.
- LabelEncoder: Etiketleri sayısallaştırır.
- TfidfVectorizer: Metinleri sayısal vektörlere dönüştürür.
- model_selection: Veri kümesini eğitim ve test verisi olarak ayırır.
- svm: Destek Vektör Makineleri (SVM) algoritması için kullanılır.
- MultinomialNB: Çok terimli Naive Bayes sınıflandırma algoritması için kullanılır.
- classification_report, confusion_matrix, accuracy_score: Sınıflandırma performansını değerlendirmek için kullanılır.
- nltk.download('stopwords'): Türkçe durak kelimelerini indirir.

2- Zemberek kütüphanesini kullanmak için JVM başlatılıyor ve gerekli sınıflar yükleniyor:

```
zemberekYol = 'zemberek-full.jar'

jp.startJVM(jp.getDefaultJVMPath(), 'ea', '-Djava.class.path=%s' % (zemberekYol),ignoreUnrecognized=True)
TC_CumleAyirici, TC_Morfoloji, TC_Duzeltici, TC_Tokenlestirici, TC_Lex = (jp.JClass(x) for x in ['zemberek.
```

- zemberekYol: Zemberek kütüphanesinin dosya yolu.
- jp.startJVM(...): Java Sanal Makinesi (JVM) başlatılır.
- TC_CumleAyirici, TC_Morfoloji, TC_Duzeltici, TC_Tokenlestirici, TC_Lex: Zemberek kütüphanesindeki gerekli sınıfların adları.
-

3- Zemberek kütüphanesi ile ilgili nesnelerin oluşturulması:

```
cumle_ayirici, morfoloji, tokenizer, yazim_kontrol = TC_CumleAyirici.DEFAULT,
TC_Morfoloji.createWithDefaults(), TC_Tokenlestirici.ALL, TC_Duzeltici(TC_Morfoloji.createWithDefaults())
girdiler, ciktilar, tip = [], [], 0
```

- cumle_ayirici: Türkçe cümleleri ayırmak için kullanılır.
- morfoloji: Türkçe metinlerdeki kelimelerin köklerini çıkarmak için kullanılır.
- tokenizer: Metni tokenlere (kelimelere) böler.
- yazim_kontrol: Türkçe metinlerde yazım hatalarını kontrol etmek ve düzeltmek için kullanılır.
- makale_sayisi, girdiler, ciktilar, tip: Sınıflandırma için gerekli olan değişkenler.

4- Token analizi için token_analiz_et fonksiyonu:

```
def token_analiz_et(token) -> bool:
    t = token.getType()
    return (t != TC_Lex.NewLine and
            t != TC_Lex.SpaceTab and
            t != TC_Lex.Punctuation and
            t != TC_Lex.RomanNumeral and
            t != TC_Lex.UnknownWord and
            t != TC_Lex.Unknown)
```

Bu fonksiyon, geçerli bir tokenin (kelimenin) Türkçe karakterler ve anlamlı kelimeler olup olmadığını kontrol eder.

5- Metin ön işleme fonksiyonu (metin_on_isleme):

```
def metin_on_isleme(text):
    tokens = tokenizer.tokenize(text)
    duzeltilmis_metin = ''
    for token in tokens:
        text = token.getText()
        # Eğer token işleme alınması gerekiyorsa ve yazım hatası varsa düzelt
        if (token_analiz_et(token) and not yazim_kontrol.check(text)):
            oneriler = yazim_kontrol.suggestForWord(token.getText())
            if oneriler:
                correction = oneriler.get(0)
                duzeltilmis_metin += str(correction)
            else:
                duzeltilmis_metin += str(text)
        else:
            duzeltilmis_metin += str(text)
    return duzeltilmis_metin
```

Bu fonksiyon, metindeki yazım hatalarını düzeltir ve düzeltilmiş metni döndürür.

6- Kök kelimeleri çıkarma fonksiyonu (kok_kelimeleri_cikar):

```
def kok_kelimeleri_cikar(duzeltilmis_metin):
    cumleler = cumle_ayirici.fromParagraph(duzeltilmis_metin)
    kok_kelimeler = []
    for cumle in cumleler:
        analiz = morfoloji.analyzeAndDisambiguate(cumle).bestAnalysis()
        for word in analiz:
            kok_kelimeler.append(word.getLemmas()[0])
    return kok_kelimeler
```

Bu fonksiyon, düzeltilmiş metindeki kelimelerin köklerini çıkarır ve bir liste olarak döndürür.

7- Veri kümesinden makalelerin okunması ve ön işleme:

```
for i, category in enumerate(dataset.label_data):
    for j in range(0, 21):
        text = category[j]
        duzeltilmis_metin = metin_on_isleme(text)
        kok_kelimeler = kok_kelimeleri_cikar(duzeltilmis_metin)
        kok_kelimeler = [e for e in kok_kelimeler if e not in (',', '.', '"', ";", ":", "?", "!",
        kok_kelimeler = [token for token in kok_kelimeler if token not in engelli_kelimeler]
        kok_kelimeler = str(kok_kelimeler)
        girdiler += [kok_kelimeler]
        ciktilar += [labels2[i]]
```

Bu döngüde, veri kümesindeki her kategoriden 21 makale okunur, ön işleme yapılır ve kök kelimeleri çıkarılır. Ardından girdiler ve çıktılar listelerine eklenir.

8- Çıktıların sayısallaştırılması:

```
Encoder = LabelEncoder()
ciktilar = Encoder.fit_transform(ciktilar)
```

LabelEncoder kullanarak, makale kategorilerini sayısal değerlere dönüştürürüz.

9- Veri kümesinin eğitim ve test verisi olarak ayrılması:

```
# Veri kümesini eğitim ve test kümesi olarak ayırma
x_deneme, x_test, y_deneme, y_test = model_selection.train_test_split(girdiler, ciktilar, test_size=0.3, random_state=70)
```

`train_test_split` fonksiyonu ile veri kümesini %70 eğitim verisi ve %30 test verisi olacak şekilde böleriz.

10- Girişlerin sayısallaştırılması:

```
tfidf_vector = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', ngram_range=(1, 2), max_features=10000)
tfidf_vector.fit(girdiler)
Train_X_Tfidf = tfidf_vector.transform(x_deneme)
Test_X_Tfidf = tfidf_vector.transform(x_test)
```

TfidfVectorizer kullanarak, girdi metinlerini sayısal değerlere dönüştürürüz. Bu, eğitim ve test verileri için yapılır.

11- Test makalesinin hazırlanması:

```
girdiMetin = input("Lütfen test etmek istediğiniz makaleyi girin:(Eğer boş ise varsayıla
if girdiMetin.strip() == "":
        girdiMetin = girdiMetinVarsayilan

tokens = tokenizer.tokenize(girdiMetin)
def token_analiz_et (token) -> bool:
    t = token.getType()
    return (t != 
                (variable) TC_Lex: JClass
                t != 
                t != TC_Lex.Punctuation and
                t != TC_Lex.RomanNumeral and
                t != TC_Lex.UnknownWord and
                t != TC_Lex.Unknown)

duzeltilmis_metin = ''

for token in tokens:
    text = token.getText()
    if token_analiz_et(token) and not yazim_kontrol.check(text):
        oneriler = yazim_kontrol.suggestForWord(text)
        duzeltilmis_metin += str(oneriler.get(0)) if oneriler else str(text)
    else:
        duzeltilmis_metin += str(text)

cumleler = cumle_ayirici.fromParagraph(duzeltilmis_metin)
kok_kelimeler = []
for cumle in cumleler:
    analiz = morfoloji.analyzeAndDisambiguate(cumle).bestAnalysis()
    for word in analiz:
        kok_kelimeler.append(word.getLemmas()[0])

engelli_kelimeler = nltk.corpus.stopwords.words('turkish')
kok_kelimeler = [e for e in kok_kelimeler if e not in (',', '.', '"', ";", ":", "?", "!"
kok_kelimeler = [token for token in kok_kelimeler if token not in engelli_kelimeler]
kok_kelimeler = str(kok_kelimeler)
girdi = [kok_kelimeler]
test = tfidf_vector.transform(girdi)
```

Kullanıcının gireceği test makalesi için ön işleme, kök çıkarma ve Tfidf dönüşümü yapılır.

12- Makale türünü döndüren fonksiyon (makaleTuruGetir):

```
def makaleTuruGetir(par):
    types = ["BİLİM&TEKNOLOJİ", "SAĞLIK","TARİH", "EKONOMİ","SPOR"]
    return f"{types[par]} MAKALESİ" if 0 <= par < len(types) else "Makale bulunamadı"
```

Bu fonksiyon, sınıflandırma sonucunda elde edilen sayısal değeri makale türüne çevirir.

13- Naive Bayes Algoritması:

```
Naive = MultinomialNB()
Naive.fit(Train_X_Tfidf, y_deneme)
skor = Naive.predict(Test_X_Tfidf)
sonuc = Naive.predict(test)
```

'MultinomialNB' kullanarak, eğitim verisi üzerinde Naive Bayes algoritmasını eğitir ve test verisi üzerinde tahminlerde bulunur.

14- Sonuçların yazdırılması:

```
print("\n")
print("SONUÇ :", makaleTuruGetir(sonuc[0]))
print("DOĞRULUK SKORU: ", accuracy_score(skor, y_test)*100)
print("\n")
```

Sonuçlar yazdırılır: Test makalesinin türü, ve doğruluk skoru sınıflandırma raporu.