Tweet Classification with Text Mining Techniques

Metin Madenciliği Teknikleriyle Tweet Sınıflandırma

Melike AKALAN
Kırıkkale Üniversitesi

ABSTRACT

Text mining can be used to see the demand for increased data usage. In accordance with the data set, which will be Logistic Regression, Naive Bayes and LSTM in this regard, as well as some future information in the text in the amount of data. Logistic Regression and Naive Bayes are probabilistic-based supervised learning. LSTM is a deep learning method with long-term memory used in natural language processing. What to do in the article, examples from the text in text mining, comparisons with other performance that can be evaluated.

Keywords: Logistik Regression, Naive Bayes, LSTM, classification, TweetTokenizer.

ÖZET

Metin madenciliği, artan veri miktarının ciddi bir boyuta geldiği günümüzün oldukça rağbet gören konusudur. Veri miktarındaki artış da metin sınıflandırma yöntemlerindeki bazı eksiklikleri beraberinde getirmiş bu sebeple Logistik Regression, Naive Bayes ve LSTM algoritmaları veri setine uygun olacak şekilde uygulanmıştır. Logistik Regression ve Naive Bayes, olasılık tabanlı denetimli öğrenme algoritmalarıdır. LSTM ise doğal dil işleme alanında kullanılan uzun vadeli hafızası olan bir derin öğrenme yöntemidir. Makalede bu algoritmaların ne olduğu, metin madenciliğinde kullanım biçimi, diğer sınıflandırma yöntemleriyle karşılaştırılarak yapılan performans testleri ve metin madenciliği çalışmalarına olan katkısı kısaca anlatılmaya çalışılmıştır.

Anahtar Kelimeler: Logistik Regression, Naive Bayes, LSTM, sınıflandırma, TweetTokenizer.

1. GİRİŞ

Günümüzde insanlar düşüncelerini, duygularını ifade etmek için çoğunlukla Twitter'ı kullanmaktadır. Twitter'da yer alan veriler(tweetler) de giderek arttığından metin madenciliği çalışmalarına konu olmuştur. Metin madenciliği; sosyal medya madenciliği, duygu analizi, yazar tespiti, kişiselleştirilmiş reklamlar, bilimsel araştırmalar gibi pek çok alanda kullanılmaktadır.

Bildiğiniz üzere veriler; tek başına anlamı olmayan varlıklardı, üzerinde işlem yapılmadıkları takdirde arşiv özelliğinde saklanabiliyorlardı; fakat bu veriler arasından aradığımız anlamlı, doğru ve geçerli verilere ulaşabilmek için birtakım aşamalardan geçerek bilgiye dönüştürülmesi gerekir. İşte bu noktada karşımıza çıkan klasik metin işleme yöntemleri; giderek hızla artan veri karşısında yetersiz kalarak, yerlerini **"olasılık tabanlı tweet sınıflandırma"** veya **"uzun vadeli bilgi saklayan sınıflandırma"** yöntemlerine bırakmışlardır.

Twitter verisi, miktar olarak çok büyük, sonsuz ve süreklidir. Veri sürekli değiştiğinden veri hakkında tahminde bulunmak zordur. Buna bağlı olarak seçilecek veri özelliklerinin de doğru bir şekilde belirlenmesi zorlaşır. Dolayısıyla geliştirilecek yöntemin ya bu değerleri otomatik belirlemesi ya da esnek bir yapıda olması gerekir. Bize de bunu **"Logistik Regression, Naive Bayes ve LSTM yöntemleri"** sağlar. Çünkü bu yöntemlerde metinlerde geçen kelimelerin ve emojilerin olasılıkları hesaplanarak ilerlenir ve geçmiş bilgileri de hatırladığı için başarılı sonuçlar elde edilir.

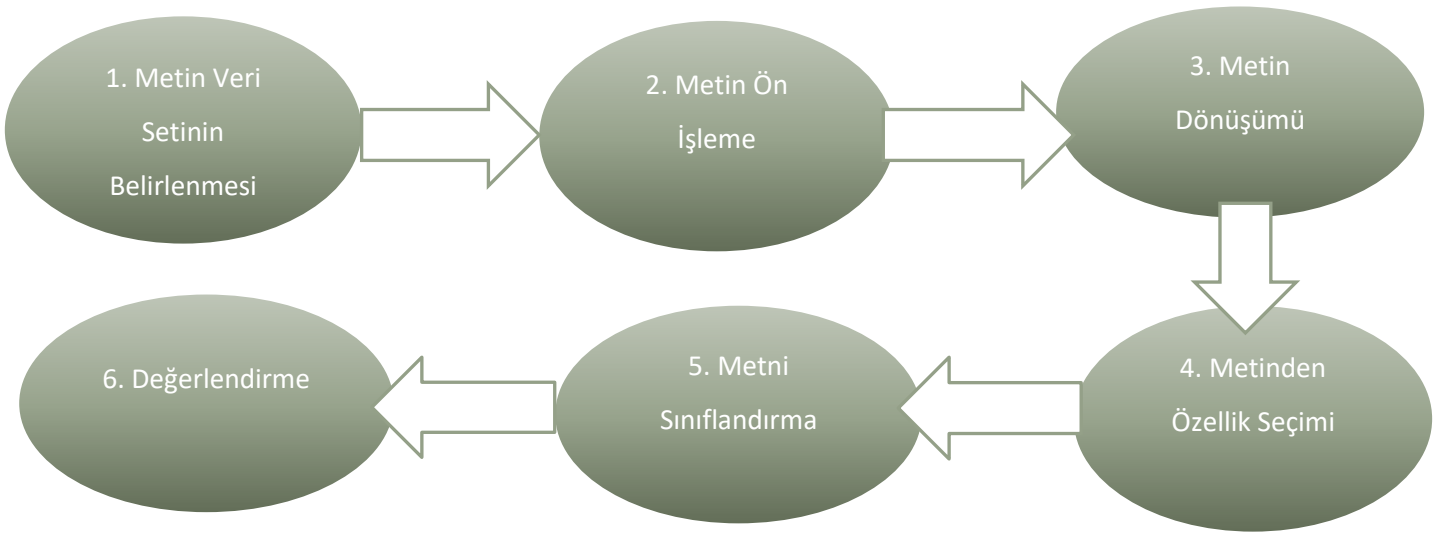
Projede 5.000'i positive 5.000'i negative olmak üzere toplam 10.000 tweetten oluşan iki ayrı .json dosyalarıyla çalışılmıştır. Bu dosyalar nltk kütüphanesi kullanılarak indirilmiştir. İndirilen dosyalarda veriler etiketli değildir yani içerisinde her bir tweet için positive ya da negative olduklarına dair herhangi bir bilgi yoktur. Bunun için ilk olarak verilerin etiketlenmesi işlemi gerçekleştirilir, bu işlem python dilinin sözlük yapısından faydalanarak gerçekleştirilir. Buradaki sözlük yapısında ilgili tweet ve yanında etiketi belirtilir. İlgili tweet "positive_tweets.json" dosyasında yer alıyorsa"1", "negative_tweets.json" dosyasında bulunuyorsa "0" olarak etiketlenir. Daha sonra bahsi geçen sınıflandırma yöntemleri kullanılarak tweetler positive ya da negative olacak şekilde sınıflandırılmıştır.

Projenin sonunda ise summary.py dosyası oluştulmuş ve burada veri seti olan positive_tweets.json, negative_tweets.json dosyalarının tamamı dataframe formatına dönüştürülmüştür. Dataframe yapısına dönüştürülen tweetleri etiketlemek için ek bir sütun oluşturulmuş ve yanlarına yine sözlük yapısındaki gibi "1" ya da "0" olma durumları eklenmiştir. Daha sonra da bu dataframe üzerinden ilgili sınıflandırma algoritmaları eğitilmiş ve sınıflandırma sonuçları, başarı oranları karşılaştırılmıştır.

2. GELİŞTİRİLEN YÖNTEMLER

Twitter verilerini sınıflandırmak için geliştirilen yöntemleri kullanmadan önce veriler üzerinde metin madenciliği teknikleri kullanılarak veri işleme hazır hale getirilir. Verinin sınıflandırmaya hazır hale gelmesi için ilk olarak verinin yani metnin içerisindeki positive/neagative olma olasılığına katkı sağlamayan kelimelerin, noktalama işaretlerinin, tekrar eden kelimelerin temizlenerek ön işleme aşamalarını tamamlaması gerekir. Ön işleme aşamalarını tamamlayan tweetler metin dönüşümü aşamasında, vektörel hale getirilerek temsil edilir. Vektörel temsili sağlanan tweetin sınıflandırılması için gerekli öznitelikler seçilir ve sınıflandırma algoritmasına verilir. Sınıflandırma işlemi gerçekleştikten sonra da değerlendirme aşamasına geçilerek algoritmaların performansı test edilir.

Bu aşamaların akış şeması aşağıdaki gibidir.



2.1 Logistik Regression

Logistik regression; eğitilen modelin ikili sınıflandırma yapmasını sağlayan denetimli makine öğrenme algoritmasıdır. Tweetin pozitif ya da negatif olma olasılığını hesaplarken sigmoid fonksiyonu kullanılır. Sigmoid fonksiyonu her tweet için 0 ile 1 arasında değer alan bir olasılık hesaplar, hesaplanan bu değere tahmin çıktısı**(y_hat)** denir. Tahmin çıktısı, sigmoid eğrisindeki threshold değerinden (0.5 ten) büyükse tweet pozitif; küçükse negatif şeklinde sınıflandırılır. Sonuç olarak sigmoid eğrisi tweetleri negatif ve pozitif olmak üzere ikiye ayırmış olur. Logistik regression da bilindiği üzere discriminative (ayırıcı) bir algoritmadır.

Logistik regression ile modeli eğitirken amacımız doğru thetaları güncelleyerek minimum maliyet(bütün tweetler için yanlış sınıflandırılma olasılıkları ortalaması) olacak şekilde veri setine eğitim yaptırmaktır. Thetalar ilk başta random bir değerle başlatılır, her iterasyonda maliyet düşürülerek güncellenir. Bu sayede minumum maliyet ile yani en düşük hata oranı ile doğru sınıflandırma gerçekleşmiş olur.

2.1.1 Logistic Regression Algoritması

Daha önce de bahsedildiği gibi herhangi bir tweet, ilgili algoritmaya girmeden önce ön işleme aşamalarını tamamlaması gerekir. Bu ön işleme aşamalarını her tweet için tek tek uygulamak yerine Logistic Regression için **utils_1.py** adında bir python modülü yazdım. Modülde tweet içerisinde yer alan gereksiz kelimeler, bazı semboller ve linkler temizlendi. Temizlenmiş tweeti parçalara ayırmak için ise nltk kütüphanesinde yer alan **TweetTokenizer** fonksiyonu kullanıldı bu sayede emojiler de birer token kabul edilerek tahmin olasılıklarına etkisi artırıldı. Ayrıca yine aynı modül içerisinde olan

build_freqs() fonksiyonu da ilgili tweet içerisinde yer alan temizlenmiş tokenların veri setinde kaçar kez geçtiğini sayarak olasılık hesaplamasında kullanılmasını sağlar.

Algoritma temelde üç aşamadan oluşmaktadır:
- ön işlemeden geçen ve vektörleştirilen tweetler için önce maliyet hesaplanır,
- tweet için gerekli olan features(özellikler) seçilir,
- son olarak da tweetin tahmin çıktısı hesaplanarak sınıflandırılır.

```
from nltk.corpus import twitter_samples                 # veri setinin yüklenmesi
from utils_1 import process_tweet, build_freqs          # ön işleme ve frekans için gerekli modüller
def gradientDescent();                                  # minumum maliyet fonksiyonu
def extract_features();                                 # tweet için gerekli özellik çıkarımı fonksiyonu
def predict_tweet();                                    # tweet in tahmin çıktısı hesaplama foksiyonu
```

2.2 Naive Bayes

Naive Bayes de yine Logistic Regression gibi modelin ikili sınıflandırma yapmasını sağlayan denetimli makine öğrenme algoritmasıdır. Algoritma tweet in pozitif ya da negatif olma olasılığını hesaplarken loglikelihood, logprior gibi olasılık değerlerinden yararlanır. Bu olasılık değerlerine göre elde edilen **p değeri**(tahmin çıktısı) elde edilir. Bu değer 0.0 dan büyükse tweet pozitif, küçükse negatif şeklinde sınıflandırılır. Naive Bayes'in Logistic Regression'dan en temel farkı, generative (gruplayıcı) bir algoritma olmasıdır. Yani ilgili tweetleri sınıflandırırken herhangi bir thresold değeriyle onları birbirinden ayırmak yerine, olasılıkları üzerinden gruplamayı tercih eder.

Tweetlerde yer alan her bir temizlenmiş token(kelime ya da emoji) bir duyguyu, bir düşünceyi ifade eder. Bu sebeple aslında her token bizim için önemlidir ve bunları Naive Bayes'in olasılık formulüne göre hesapladıktan sonra elde edilen olasılık değerlerine göre tweetleri gruplandırarak sınıflandırma işlemi gerçekleştirilir.

2.2.1 Naive Bayes Algoritması

Logistic Regression algoritmasında bahsettiğim ön işleme ve frekans hesaplama için yazılan python modülünü burada Naive Bayes algoritmasına uyarlayarak **utils_2.py** adında oluşturdum.

Algoritma temelde üç aşamadan oluşmaktadır:
- ön işlemeden geçen ve vektörleştirilen tweetler önce frekansları(içerisindeki tokenların bulunma sayısı) ve etiketleriyle(tokenlar pozitifse:1, negatifse: 0 durumu) temsil edilir,
- eğitim verisi için ayrılan tweetler algoritmayla eğitilir,
- son olarak da tweetin tahmin çıktısı hesaplanarak sınıflandırılır.

```
from nltk.corpus import twitter_samples          # veri setinin yüklenmesi
from utils_2 import process_tweet, lookup        # ön işleme ve frekans için gerekli modüller
def count_tweets();                              # tweet in frekansları ve etiketiyle temsili
def train_naive_bayes();                         # modelin eğitim fonksiyonu
def naive_bayes_predict();                       # modelin tahmin çıktısı fonksiyonu
```

2.3 LSTM

LSTM(Long Short-Term Memory) adından da anlaşılacağı üzere hafızasında uzun süreli bilgi saklayabilen, tekrarlayan sinir ağı(RNN) mimarisidir. LSTM'yi anlamak için ilk olarak derin öğrenme yöntemi olan RNN mimarisini anlamamız gerekir. RNN ilgili kelimenin positive/negative olma olasılığını hesaplarken bir önceki kelimenin olasılığını kullanır ve ürettiği her çıkışın bir önceki adıma bağlı olarak ilerlemesini sağlar. Önceki adımlarda hesaplanan sonuçları hafızasında tutmaya çalışır, bu sayede onların da sınıflandırmaya dahil etmiş olur. RNN'ler kısa, sade, anlaşılır metinlerde iyi sonuçlar üretir fakat uzun, karmaşık yapıdaki metinlerde gördüklerini unutabilir ve kısa süreli bir hafızaya sahip olup başarısız olabilirler.

LSTM, bilgileri daha iyi saklayarak, standart RNN'in kısa vadeli hafıza problemini çözmek için tasarlanmıştır. LSTM, RNN'den farklı olarak **unutma kapısını** kullanılır ve bu kapıda hangi olasılıkların unutulacağı ya da saklanacağını belirlenir. Bu belirleme işleminde, önceki adımdan gelen olasılık ve mevcut olasılık sigmoid fonksiyonuna verilir. Sigmoid çıktısı 0(sıfır) olan olasıklar unutulur, 1 olan olasılıklar sınıflandırma işlemine dahil olur. Böylece metin sınıflandırma işleminde daha başarılı sonuçlar elde etmemizi sağlar.

3. PERFORMANS TESTLERİ VE TARTIŞMA

3.1 Sınıflandırma Başarısını Ölçme

Sınıflandırma başarısını ölçmek için Accuracy, Precision, Recall, F1-score, MAE ve ROC-curve parametreleri kullanılmıştır.

- **Accuracy**, modelde doğru tahmin edilen tweetlerin toplam tweet sayısına oranıdır.
- **Precision**, positive olarak tahmin edilen tweetlerin gerçekten kaçının positive olduğudur.
- **Recall,** positive olarak tahmin etmemiz gereken tweetlerin kaç tanesinin positive olarak tahmin edildiğidir.
- **F1-score,** precision ve recall değerlerinin harmonik ortalamasını gösteren bir değerdir.
- **MAE,** modelin sınıflandırma işleminde yaptığı ortalama hatadır.
- **ROC-curve**, modelin sınıflandırma başarısını gösteren eğridir. Eğri altında kalan alan büyüdükçe modelin başarısı artmaktadır.

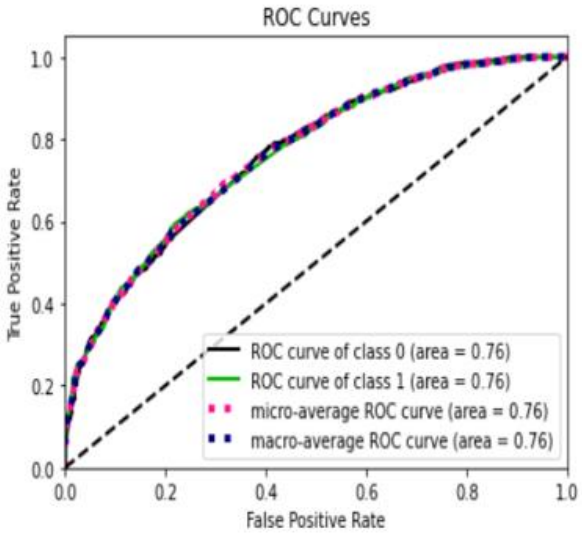
3.2 Test Sonuçları

Logistic Regression, Naive Bayes ve SVM algoritmalarında tweetleri tokenlaştırmak için **word_tokenize** kullanıldığında accuracy değerleri 0.68 ve 0.67 olmaktadır. Ortalama hata değerleri ve ROC eğrileri de birbirlerine benzer ortalama sonuçlar göstermektedir.

```
Lojistik Regression Accuracy Score:  0.6865

              precision    recall  f1-score   support

           0       0.66      0.76      0.71       997
           1       0.72      0.62      0.66      1003

    accuracy                           0.69      2000
   macro avg       0.69      0.69      0.69      2000
weighted avg       0.69      0.69      0.68      2000
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, predictions_LR)
```

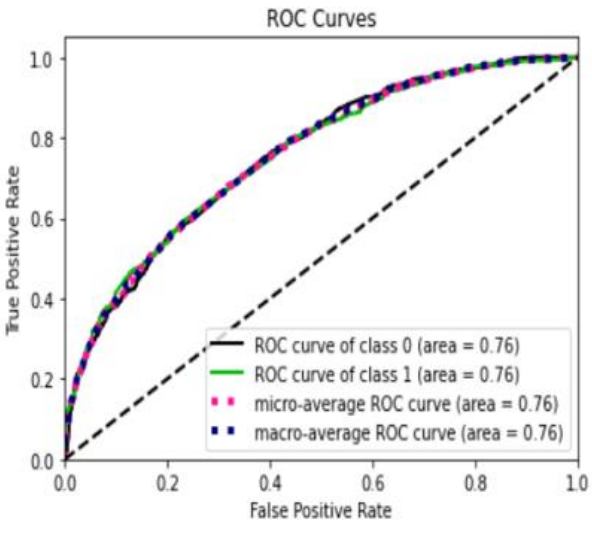
```
0.3135
```




```
Naive Bayes Accuracy Score:  0.6795

              precision    recall  f1-score   support

           0       0.65      0.77      0.70       997
           1       0.72      0.59      0.65      1003

    accuracy                           0.68      2000
   macro avg       0.69      0.68      0.68      2000
weighted avg       0.69      0.68      0.68      2000
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, predictions_NB)
```

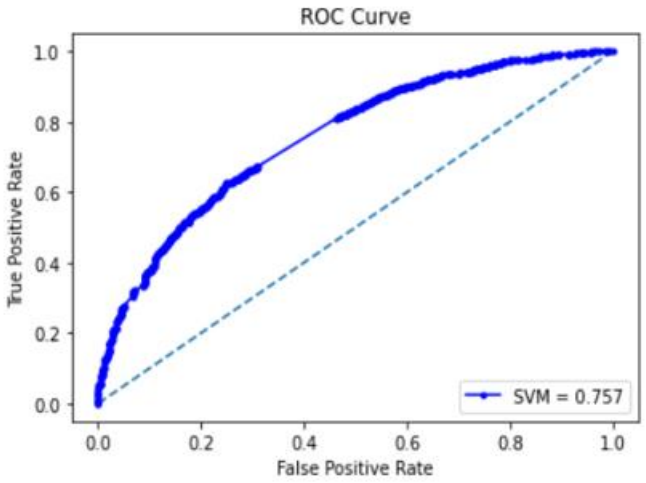
```
0.3205
```



```
SVM Accuracy Score:  67.95
              precision    recall  f1-score   support

           0       0.65      0.76      0.70       997
           1       0.72      0.60      0.65      1003

    accuracy                           0.68      2000
   macro avg       0.68      0.68      0.68      2000
weighted avg       0.68      0.68      0.68      2000
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, predictions_SVM)
```

```
0.3205
```



Logistic Regression ve Naive Bayes algoritmalarında tweetleri tokenlaştırmak için **TweetTokenizer** kullanıldığında ise accuracy değerleri 0.995 ve 0.994 değerlerine çıkmaktadır.

```
def test_logistic_regression(test_x, test_y, freqs, theta):

    # predictions için liste
    y_hat = []

    for tweet in test_x:
        y_pred = predict_tweet(tweet, freqs, theta)
        if y_pred > 0.5:
            y_hat.append(1)
        else:
            y_hat.append(0)

    # Yukarıdaki uygulamayla, y_hat bir listedir, ancak test_y (m, 1)
    # '==' operatörünü kullanarak karşılaştırmak için her ikisi de tek

    accuracy = (y_hat==np.squeeze(test_y)).sum()/len(test_x)

    return accuracy
```

```
tmp_accuracy = test_logistic_regression(test_x, test_y, freqs, theta)
print(f"Logistic regression model's accuracy = {tmp_accuracy:.4f}")
```

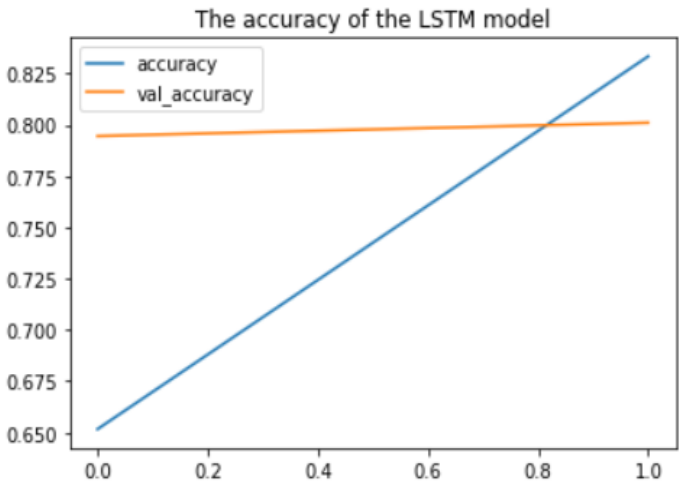
```
Logistic regression model's accuracy = 0.9950
```

```python
def test_naive_bayes(test_x, test_y, logprior, loglikelihood):

    accuracy = 0

    y_hats = []
    for tweet in test_x:
        if naive_bayes_predict(tweet, logprior, loglikelihood) > 0
            y_hat_i = 1   #tahmin olasılığı 0 dan büyükse 1 ata
        else:
            y_hat_i = 0   #tahmin olasılığı 0 dan küçükse 0 ata

        # tahmin edilen sınıfı y_hats listesine ekle
        y_hats.append(y_hat_i)

    # hata, y_hats ve test_y arasındaki farkların mutlak değerleri
    error = np.mean(np.absolute(y_hats-test_y))
    accuracy = 1-error

    return accuracy
```

```python
print("Naive Bayes doğruluğu = %0.4f" %
      (test_naive_bayes(test_x, test_y, logprior, loglikelihood)))
```

```
Naive Bayes doğruluğu = 0.9940
```

LSTM algoritmasında tweetleri tokenlaştırmak için tensorflow kütüphanesinin **Tokenizer** fonksiyonu kullanıldığında ise 0.82 olmaktadır, hata değerleri de buna bağlı olarak azalmaktadır.



4. SONUÇ

Tweetleri “TweetTokenizer” fonksiyonu kullanarak sınıflandırdığımızda başarı oranı artar çünkü bu fonksiyon emojileri de birer token kabul ettiğinden onların da olasılık hesaplamalarına katılmalarını sağlar. Tweetleri “word_tokenize” kullanarak sınıflandırdığımızda ise başarı oranı düşer çünkü bu yöntem emojileri birer punctuation(noktalama işaretleri) olarak kabul edip, olasılık hesabına katılmaları engellemektedir.

Bütün algoritmalar arasında ise en başarılı sonucu 0.995 accuracy değeriyle Logistic Regression sağlamıştır. Bunun sebebi de gradyan inişi ile yavaş yavaş ağırlıkları güncelleyerek minumum hatayı bulması ve doğru öznitelikleri seçmesinden kaynaklanır.

Projenin Metin Madenciliği Çalışmalarına Olan Katkıları:

- ✓ Logistic Regression, Naive Bayes ve LSTM algoritmalarının metin madenciliğine uyarlanması,
- ✓ Tokenlaştırma yöntemlerinin metin sınıflandırmasına etkisinin incelenmesi,
- ✓ Sınıflandırma başarısının farklı algoritmalar arasında karşılaştırılarak veri setine uygun sınıflandırıcı ve token yönteminin belirlenmesidir.

Bu projenin Türkiye'de ve dünyada benzerleri adlarını sıkça duyduğumuz "twitter duygu analizi, twitter doğal dil işleme, sosyal medya madenciliği" olarak mevcuttur. Daha büyük daha karmaşık, düzensiz dağımlı(positive ve negative tweet sayıları eşit oranda olmayan) veri setinde accuracy değerleri yine 0.995 gibi iyi sonuçlar görmek için karşımıza Twitter API çıkmaktadır. Twitter API'nin sunmuş olduğu büyük veri kaynaklarını Apache Spark teknolojisiyle kullanarak daha iyi sonuçlar gözlemlemek mümkündür.

5. KAYNAKLAR

[1] https://dergipark.org.tr/tr/pub/ngumuh/issue/35079/383709

[2]https://www.researchgate.net/publication/338363067_NaiveBayes_Classifier_on_Twitter_Sentiment_Analysis_BPJS_of_HEALTH

[3] https://dergipark.org.tr/en/pub/humder/issue/56545/772929

[4] https://dergipark.org.tr/tr/pub/estudambilisim/issue/53654/676052

[5] https://dergipark.org.tr/tr/download/article-file/676836

[6] https://dergipark.org.tr/en/download/article-file/1400473

[7] https://dergipark.org.tr/tr/download/article-file/493077

[8] http://web.firat.edu.tr/mbaykara/ubmk3.pdf

[9]https://docplayer.biz.tr/57030101-Metin-madenciligi-yontemleri-ile-twitter-duygu-analizi-twitter sentiment-analysis-using-text-mining-methods.html

[10] https://dergipark.org.tr/tr/download/article-file/1312721

[11] https://dergipark.org.tr/tr/download/article-file/1459231

[12] https://www.youtube.com/c/inzvateam/playlists

[13] Coursera – NLP Specialization Course