



T.C.

**DÜZCE ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
BÖLÜMÜ**

**YANILTICI İÇERİK TESPİT
ALGORİTMASI**

BM401

Enes Bilal Şeker - 171001020

Özet

Bu projenin amacı günümüzde sosyal medya kullanımının aşırı yaygınlaşmasından doğan ve son yıllarda büyük sorunlardan biri haline gelen sahte veya yanlış bilgi yayılmasının olabildiğince önüne geçilebilmesidir. Bu hedef doğrultusunda, paylaşılan bilginin doğru veya yanlış olduğunun analizini Doğal Dil İşleme (Natural Language Processing - NLP) yardımı ile yapan bir algoritma geliştirildi. Algoritma daha önce doğru/sahte olarak etiketlenmiş veri seti ile eğitildi, eğitim haberin/tweetin içerdiği kelimeler baz alınarak yapıldı. Eğitim sonucunda test setinde yapılan tahminde optimum doğruluk oranını yakalamak için scikit-learn adlı Python kütüphanesinden 3 farklı fonksiyonu kullanıldı. Bunlar; Logistic Regression (Lojistik Regresyon), Decision Tree Classifier (Karar Ağacı Sınıflandırma), Random Forest Classifier (Rastgele Orman Sınıflandırma).

1. GİRİŞ

Sosyal medyanın bu kadar erişilebilir olması faydalı bilginin olduğu kadar zararlı bilgilerin de çok hızlı bir şekilde yayılmasına yol açtı. Günlük hayatımızın arasında açıp baktığımız, sadece 10-20 saniyede okuyarak bir beğeni bıraktığımız bu metinler aslında gerçekten de hakikat mi? [1] Tabi ki okuduğumuz her habere, beğeni bıraktığımız her tweete bu soruyu sormayacak kadar kendi hayatlarımızla meşgulüz. Bu sebepten artık bu hakikat kontrolü (fact checking) işini o konuda uzmanlaşmış bir yapay zeka algoritmasına bırakmak bizim için en hızlısı ve güvenlisi olacaktır.

Bu projede temel hedef yapay zekanın bu pratikliğini kullanarak sosyal medya kullanıcılarına daha güvenli bir deneyim sunmaktır.

Günümüzde Python diliyle ve kütüphaneleriyle bu tarz projeler geliştirmek için çok fazla imkan mevcuttur. Bu projede en çok kullanılan Python kütüphanelerinden biri olan Scikit-learn'den [2] yararlanılmıştır. İçinde tahmine dayalı veri analizi yapabilmek için gerekli bütün araçları bulunduran, erişilebilirliği yüksek bir kütüphanedir.

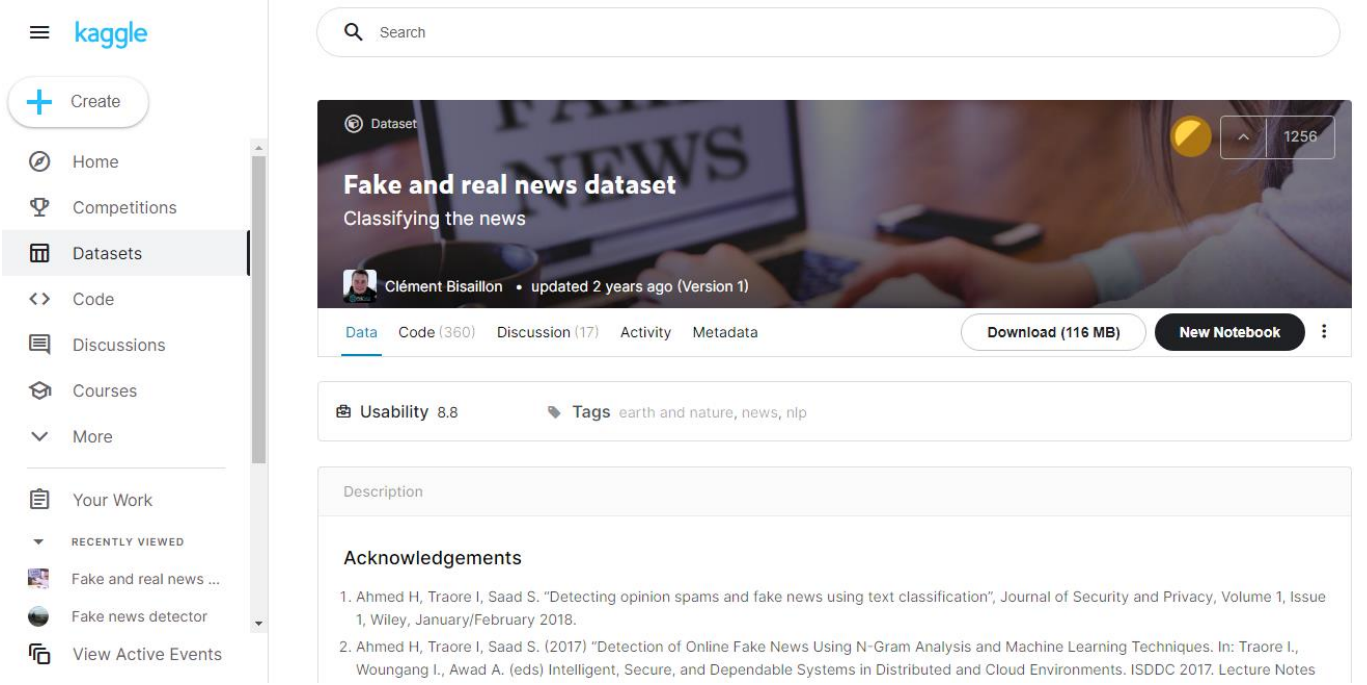
Bu projede bir Doğal Dil İşleme (Natural Language Processing - NLP) [3] yöntemi olan kelime segmentasyonu (Tokenization) [4] kullanılmıştır. Bu yaklaşımda kesintisiz bir metin yığını ayrı kelimelere dönüştürülür. Bu projede metin analizi yapılıyor fakat algoritmalar ham, çoğu değişken uzunluktaki metin belgeleri ile beslenemez. Algoritma işlemek için sabit boyutlu ve sayısal özellik gösteren vektörler bekler. Bu sebepten elimizdeki ham metin verisini, sayısal veriye dönüştürmek gerekir.

2. MATERYAL VE YÖNTEM

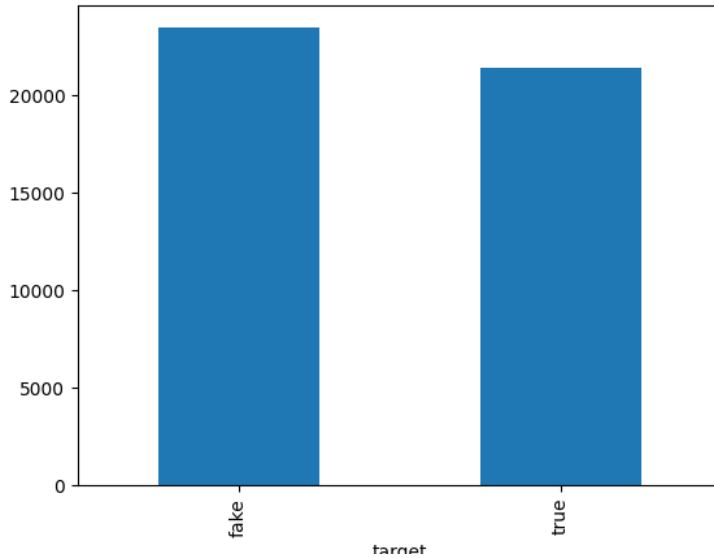
2.1. Kullanılan Veri Seti

Veri seti kişilerin topladıkları verileri serbestçe diğer insanların kullanımına sunduğu web hizmeti olan Kaggle'dan [5] alınmıştır.

Veri seti iki ayrı excel dosyasından oluşur. Excel dosyalarından biri doğru (true.csv) içerikleri diğeri ise sahte (fake.csv) içerikleri tutuyor.



The screenshot shows the Kaggle interface for the 'Fake and real news dataset'. The left sidebar contains navigation links like 'Create', 'Home', 'Competitions', 'Datasets', 'Code', 'Discussions', 'Courses', and 'More'. The main content area displays the dataset title 'Fake and real news dataset' with a subtitle 'Classifying the news'. It shows the creator 'Clément Bissillon' and 'updated 2 years ago (Version 1)'. There are buttons for 'Download (116 MB)' and 'New Notebook'. Below this, the 'Usability' is 8.8 and 'Tags' are 'earth and nature, news, nlp'. The 'Description' section includes 'Acknowledgements' for two papers.



Veri setlerinde doğru için örnek sayısı 21418, sahte için ise 23503 satır kadardır. Her satır veri için 4 sütun veri bulunur. Bunlar; başlık (title), metin (text), konu (subject) ve tarihtir (date).

Projede gerek olmadığı için veri eğitim setine dönüştürülmeden önce tarih ve başlık sütunları verinin kullanacağımız halinden çıkartıldı. Yine sadece veri setindeki kelimeler ile ilgilendiğimiz için, metin noktalama işaretlerinden arındırıldı. Ayrıca veri analizimizi metinde tekrar eden kelimelere göre yapacağımız için, İngilizce dilinde kullanılan a, an, is, the, are gibi tek başına bir anlam ifade etmeyen kelimeler metinden çıkarıldı.

```
subject
Government News    1570
Middle-east        778
News                9050
US_News             783
left-news           4459
politics            6841
politicsNews        11272
worldnews           10145
Name: text, dtype: int64
```

Veri setlerinin içindeki haber metinlerinin konu başlıkları bu şekilde dağılıyor.

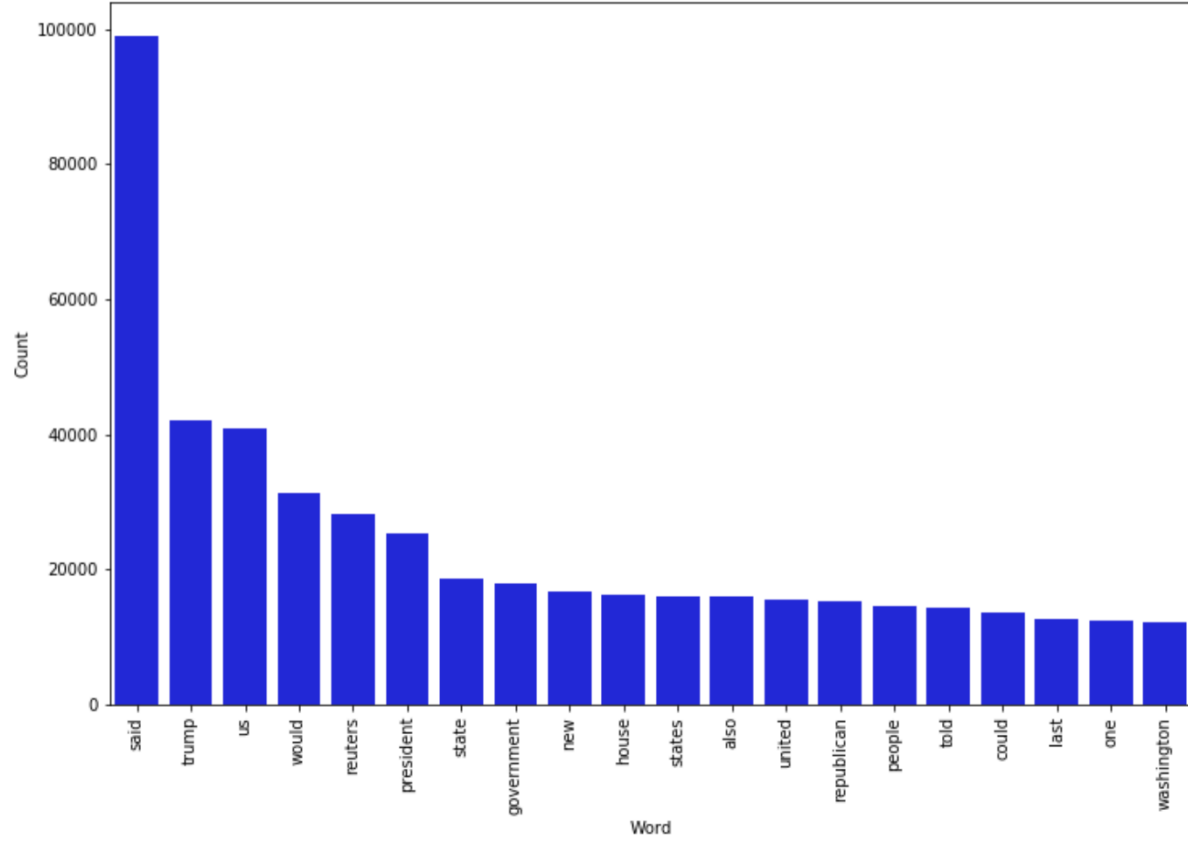
Elimizdeki 2 ayrı .csv dosyasında bulunan veri setleri algoritma eğitilmeden önce kendi içlerinde %20 test %80 eğitim verisi olarak ikiye bölündü. Ayrılan bu verilerle algoritma eğitildikten sonra, algoritmanın daha önce görmediği bir veri üzerinde ne kadar başarı ile tahmin yapabildiğinin testi yapıldı.

2.2. YANILTICI İÇERİK TESPİT ALGORİTMASI

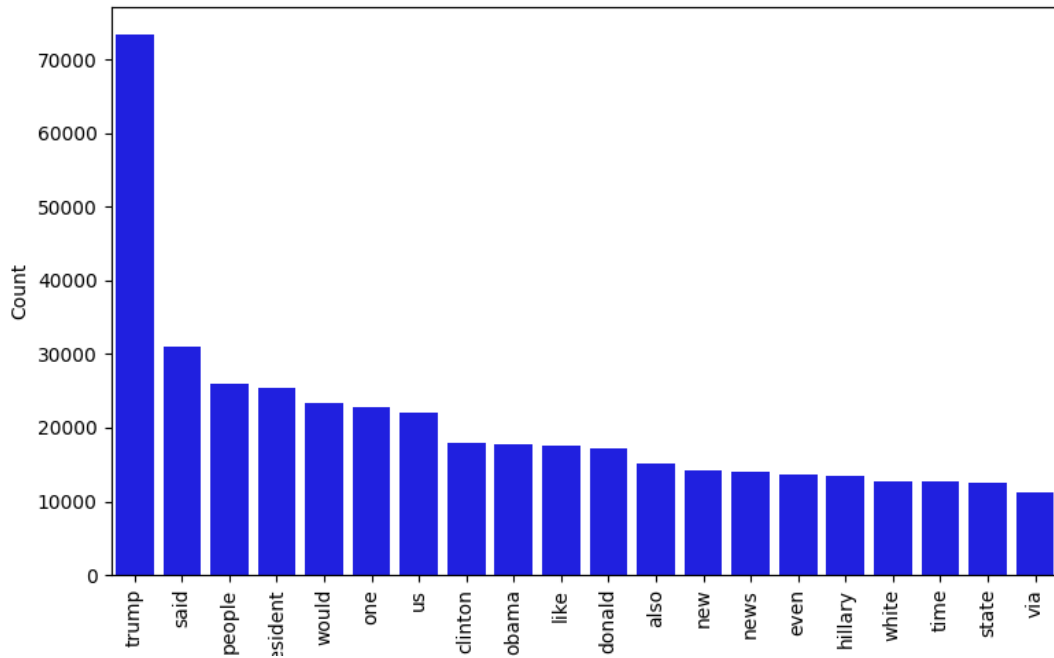
Girişte değindiğim tokenize işlemi için scikit-learn kütüphanesinin CountVectorizer [6] ve TfidfTransformer [7] fonksiyonları kullanılmıştır.

Daha önce bahsedildiği gibi algoritmalar işlemek için sayısal değerler beklerler. Bu yüzden CountVectorizer ile metindeki kelimelerin hangi sıklıkla geçtiğini sayıp onu bir matrise atayacağız. Bu fonksiyon metinde geçen her bir tokene (yani kelimeye) id olarak bir integer değer atar. Böylece aynı id ile tekrar tekrar karşılaşması durumunda onları daha rahat sayar. Bunları yaparken kelimedenden kelimeye atladığını anlatmak için boşluk karakteri ayıraç olarak verilir. Böylece tekrarlanma sıklığını belirlediği kelimeleri sayısal vektörlere dönüştürür ve kullanılmaya hazır hale getirir.

TfidfTransformer (term-frequency times inverse document-frequency) [8] kullanmamızdaki hedef ise belirli bir metnin küçük bir kısmında, çok sık meydana gelen ve dolayısıyla deneysel olarak daha az bilgilendirici olan belirteçlerin etkisini azaltmaktır.



Doğru içeriklerde en sık geçen kelimeler.



Sahte içeriklerde en sık geçen kelimeler.

Projemizde metin analizi yapıldığından ağırlıklı bahsettik bu analizleri yapmak için yine bir Python kütüphanesi olan nltk'den (Natural Language Toolkit) [9] yararlanıldı. İnsan dili verisinin analizini en iyi şekilde yapmak için içinde kullanışlı araçlar barındıran bir kütüphanedir. Ayrıca nltk'den kütüphanesinin stingleri tokenize etmesi yani bir string metni kelime kelime substringlere daha sonra implemente edeceğimiz metinde tekrar eden ifadeleri yakalamakta kullanılır.

Verimizi de hazırladıktan sonra artık algoritmayı besleyip tahmin yapmak üzere 3 farklı yönteme başvuruldu. Bunlar; Logistic Regression (Lojistik Regresyon) [10], Decision Tree Classifier (Karar Ağacı Sınıflandırma) [11], Random Forest Classifier (Rastgele Orman Sınıflandırma) [12].

Logistic Regression (Lojistik Regresyon), bir verinin 0 veya 1 olduğunu yani var ya da yok olduğunu ele alarak çalışır. Elindeki veriye lineer bir doğru çizmek yerine S şeklinde bir lojistik fonksiyon çizer, bu eğimli doğru 0'dan 1'e doğru ilerler. Bu yöntem sınıflandırma problemleri için kullanılır. Yani bir girdinin doğru olmak ihtimali %50'den fazla ise o girdi 1 yani doğru sınıfına atılır. Aksine doğruluk ihtimali %50'den az olan girdiler de 0 yani yanlış sınıfına atılır.

Decision Tree Classifier (Karar Ağacı Sınıflandırma), veriye sorular sorarak ağaç yapısında ilerlemesini ve önüne çıkan koşul/sorulara verdiği cevaplarla bir çıktıya ulaşmasını amaçlayan yaklaşımdır. Binary bir ilerleme ile girdileri sınıflandırır. Girdi en ağacın en uç kısmı olan yaprağa gelen kadar sorgulanması devam eder.

Random Forest Classifier (Rastgele Orman Sınıflandırma), halihazırda bulunan veri setinin satırlarından rastgele bir şekilde seçilen veriler ile bootstrapped veri seti oluşturulur. Daha sonra örnekler de rastgele bir şekilde kullanılarak birden fazla ağaç oluşturulur. Daha sonra veriyi bütün bu oluşturulan ağaçlardan geçirilir ve sınıflandırma ona göre yapılır.

3. SONUÇLAR

Kullanılan 3 yöntemde de birbirine yakın doğruluk sonuçları bulundu;

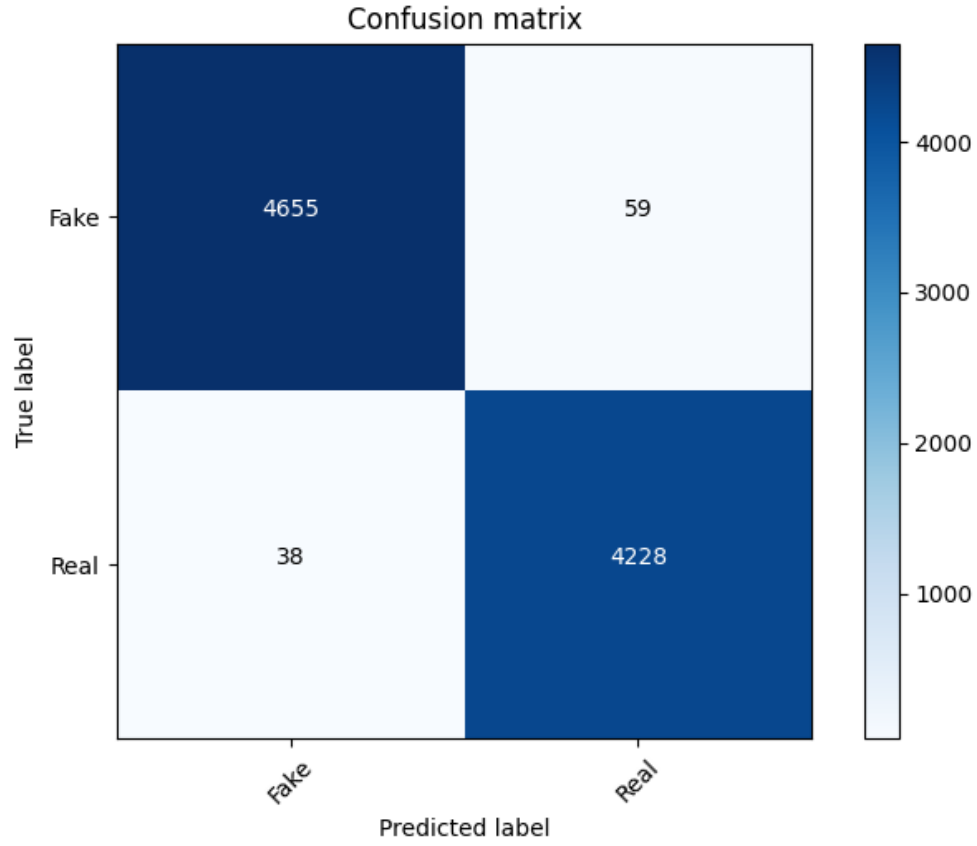
Logistic Regression (Lojistik Regresyon), yöntemi kullanılarak eğitimi yapılan algoritma, test seti için 98.98% doğruluk oranı vermiştir.

Decision Tree Classifier (Karar Ağacı Sınıflandırma), yöntemi kullanılarak eğitimi yapılan algoritma, test seti için 99.72% doğruluk oranı vermiştir. En yüksek doğruluk oranı bu yöntem ile eğitilen algoritma ile alınmıştır.

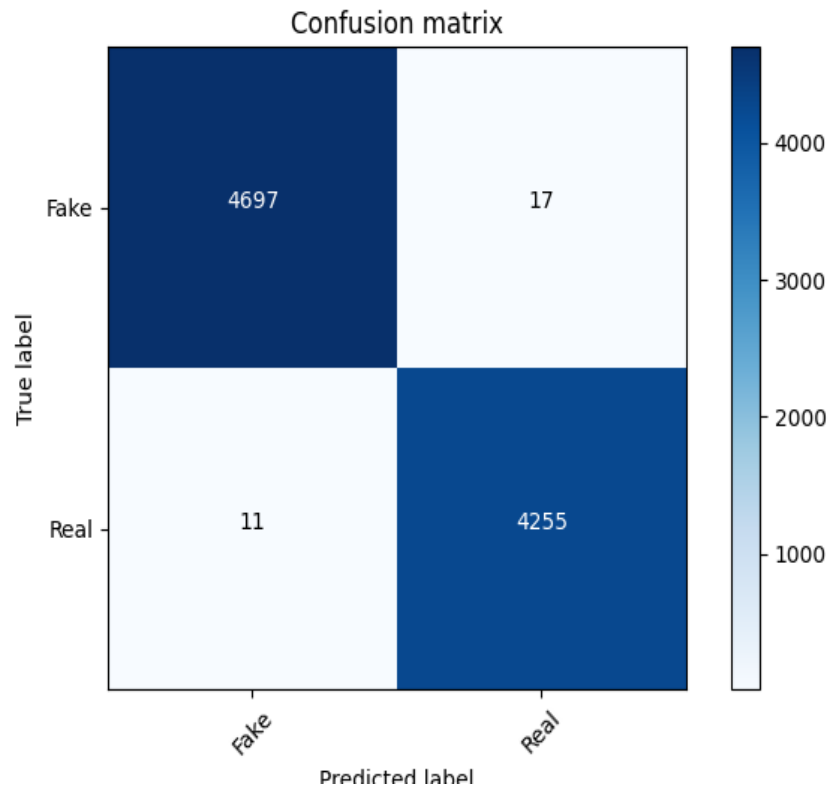
Random Forest Classifier (Rastgele Orman Sınıflandırma), yöntemi kullanılarak eğitimi yapılan algoritma, test seti için 99.28% doğruluk oranı vermiştir.

```
Logistic Regression Accuracy: 98.98%
Confusion matrix, without normalization
Decision Tree Classifier accuracy: 99.72%
Confusion matrix, without normalization
Random Forest Classifier Accuracy: 99.28%
Confusion matrix, without normalization
```

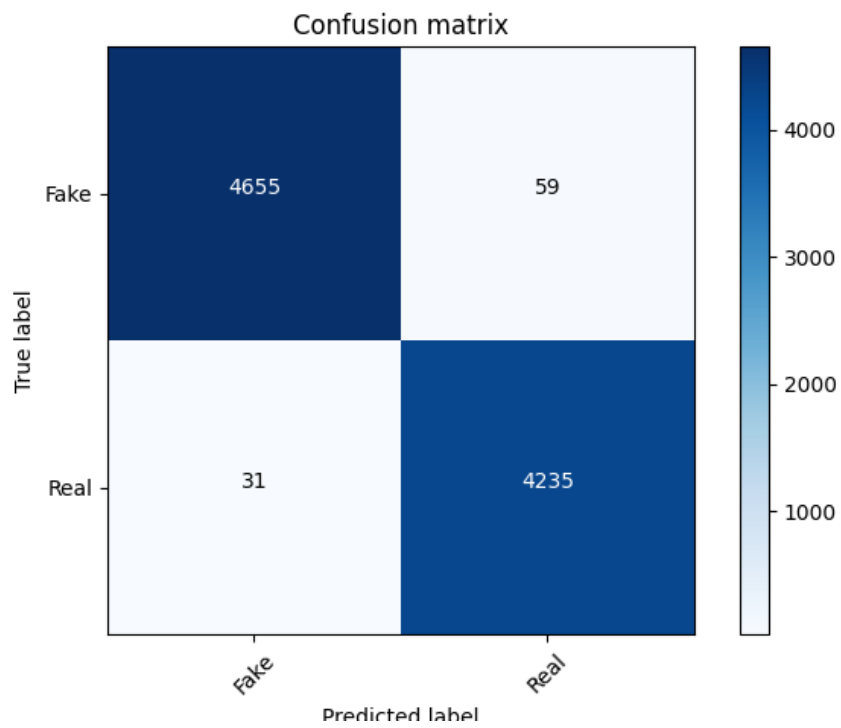
Ayrıca algoritmanın başarısını göstermek için Karışıklık Matrislerinden (Confusion Matrix) [13] yararlanılmıştır. Karışıklık Matrisleri sayesinde bir sınıflandırma yönteminin, gerçek değerleri önceden bilinmekte olan bir dizi test verisi üzerinde gösterdiği performansı tanımlamak için sıklıkla kullanılan bir tablodur.



Logistic Regression Confusion Matrix



Decision Tree Classifier Confusion Matrix



Random Forest Classifier Confusion Matrix

REFERANSLAR

- 1- <https://www.bbc.com/news/technology-53425822>
- 2- <https://scikit-learn.org/stable/index.html>
- 3- <https://www.ibm.com/cloud/learn/natural-language-processing>
- 4- https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
- 5- <https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset>
- 6- https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
- 7- https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
- 8- <http://www.tfidf.com/>
- 9- <https://www.nltk.org/>
- 10- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- 11- <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
- 12- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- 13- https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html