



**T.C.
KIRIKKALE ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ**


E-MAIL SPAM TESPİTİ


**170205022 BERİL DİNDAR
170205062 AYŞE AYBİLGE MURAT**

İÇİNDEKİLER

KOD LİSTESİ	3
TABLolar LİSTESİ	7
PROJE AÇIKLAMASI	9
SONUÇ	16
KAYNAKÇA	17

KOD LİSTESİ

 jupyter PROJE2 Last Checkpoint: 04.04.2020 (autosaved)

 Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

```
In [*]: import os
from shutil import copyfile #kaynak dosya içeriğini hedef dosyaya kopyalar.
import random
import nltk

abc = ['ham', 'spam'] #ham ve spam kelimelerinin olduğu listeyi oluşturduk
for x in abc: #kelimeleri bulunduğu kadar dönecek
    for y in range(6): #6 tane veri seti dosyamız olduğu için
        path = r"C:\Users\beril\OneDrive\Masaüstü\Proje2\enron"+str(y+1)+"\x+" #veri setimizin bulunduğu path ile 6 klasörü
        #tek tek döndürüyoruz.
        percent_70 = (0.7)*len(os.listdir(path)) # verinin yüzde 70ini aldık
        a = []
        for j in range(len(os.listdir(path))): #yolun uzunluğu kadar döndürdük
            a.append(j) # döngüdekileri a listesine ekledik.
            b = random.sample(a, int(percent_70)) # a listesine eklediğimiz benzer verilerin yüzde 70ini b parametresine aktardık.
            for i in b:
                copyfile(path+"\os.listdir(path)[i]", r"C:\Users\beril\OneDrive\Masaüstü\Proje2\trainenron/"+os.listdir(path)[i])
            for v in range(len(os.listdir(path))): # b parametresine aktardığımız verileri test ve train verisi olmak üzere
                #ayırduğumuz verinin döndürüldüğü c listesini oluşturduk.
                c = os.listdir(path)[v]
                train_path = r"C:\Users\beril\OneDrive\Masaüstü\Proje2\trainenron/"
                h = os.listdir(train_path)
                if c not in h: # eğer c parametresinin içinde train(eğitilecek) veri kalmadıysa c listesi test verisidir.
                    copyfile(path+"\c", r"C:\Users\beril\OneDrive\Masaüstü\Proje2\testenron/"+c)
```

Train & Test

Separate_train_and_test.py x Get_confusion_matrix.py x top_10_words.py x Average_word_count.py x

```
1 import os
2 import numpy as np
3 import nltk
4 from nltk.corpus import stopwords
5 from collections import Counter
6 from sklearn.model_selection import train_test_split #Veri kümesini ikiye bölmek için kullanıyoruz.
7 from sklearn.naive_bayes import MultinomialNB #çok terimli modeller için sınıflandırıcı
8 from sklearn.metrics import confusion_matrix #Bir sınıflamanın doğruluğunu değerlendirmek için karışıklık matrisini hesaplar.
9 from sklearn.svm import LinearSVC #Doğrusal Destek Vektör Sınıflandırması.
10 import warnings
11 from sklearn.exceptions import ConvergenceWarning #yakınsama hatasını görmemek için import ettim
12 def make_Dictionary(path): #3000 kelimenin seçildiği bir sözlük oluşturduğumuz fonk.
13     emails = [os.path.join(path,f) for f in os.listdir(path)] #burası mail kadar listeye bağlantıları döndürmeye yarar ve f döndüreceği sayı demek
14     all_words = [] #tüm kelimeler için bir liste oluşturduk
15     for email in emails: #email kadar mail döngüsüne soktuk
16         with open(email, encoding='latin1') as m:
17             content = m.read() #mailleri okudu
18             all_words += nltk.word_tokenize(content) #boşluk ve noktalama işaretlerine göre ayırır
19     dictionary = [word for word in all_words if word not in stopwords.words('english')]
20     dictionary = [word.lower() for word in dictionary if word.isalpha()] #Lower küçük harfe çevirir, isalpha karakterlerin alfabe var olup olmadığını kontrol eder
21     dictionary = Counter(dictionary) #sayıcıyla sözlüğü döndürdük.
22     dictionary = dictionary.most_common(3000) #en yaygın kelimeden en az kelimeye kadar döndürdük kelimelerin içinde 3000 kelimenin kaç kere tekrar ettiğini bulduk
23     return dictionary
```

Get_confusion_matrix (1) x

```
[[4810 155]
 [ 108 5037]]
[[4838 127]
 [ 77 5068]]
```

Confusion Matrix

```
Separate_train_and_test.py x Get_confusion_matrix.py x top_10_words.py x Average_word_count.py x
23 return dictionary
24 warnings.filterwarnings("ignore", category=DeprecationWarning) #önleyecği hatanın gözükeceği yer
25 def extract_features_train(path):
26     docID = 0
27     features_matrix = np.zeros((23594,3000)) #belirtilen değerler kadar 0lardan oluşan bir matris oluşturur.
28     labels = np.zeros(23594) #[23594 tane yanyana 0 değerli matris]
29     emails = [os.path.join(path,f) for f in os.listdir(path)]
30     for mail in emails:
31         with open(mail, encoding="latin1") as m:
32             all_words = []
33             for line in m: #içerik kadar satır döndürüyoruz
34                 words = line.split() #satırları bölüp kelimeleri bulduk
35                 all_words += words #kelimeleri all_words listesine ekledik
36             for word in all_words: #listedeki kelimeleri döndürüyoruz
37                 wordID = 0
38                 for i,d in enumerate(dicti): #i arttırma d ögeyi yakalama Enumerate yinelenabilir ögeye bir sayaç ekler ve bunu bir numaralandırma nesnesi biçiminde döndürür.
39                     if d[0] == word: #yakalanan öge kelimeyse
40                         wordID = i
41                         features_matrix[docID,wordID] = all_words.count(word) #dosya, kelime matrisini listedeki saydığımız kelimelerle eşitledik.
42     labels[docID] = int(mail.split(".")[2] == 'spam') #?
43     docID = docID + 1 #her döngüde +1
44     return features_matrix,labels
45 warnings.filterwarnings("ignore", category=ConvergenceWarning)

Get_confusion_matrix (1) x
[[4810 155]
 [ 108 5037]]
[[4838 127]
 [ 77 5068]]
```

Confusion Matrix

```
Separate_train_and_test.py x Get_confusion_matrix.py x top_10_words.py x Average_word_count.py x
45 warnings.filterwarnings("ignore", category=ConvergenceWarning)
46 def extract_features_test(test_dir):
47     docID = 0
48     features_matrix = np.zeros((10110,3000))
49     labels = np.zeros(10110)
50     emails = [os.path.join(test_dir,f) for f in os.listdir(test_dir)]
51     for mail in emails:
52         with open(mail, encoding="latin1") as m:
53             all_words = []
54             for line in m:
55                 words = line.split()
56                 all_words += words
57             for word in all_words:
58                 wordID = 0
59                 for i,d in enumerate(dicti):
60                     if d[0] == word:
61                         wordID = i
62                         features_matrix[docID,wordID] = all_words.count(word)
63     labels[docID] = int(mail.split(".")[2] == 'spam')
64     docID = docID + 1
65     return features_matrix,labels
66
67 path = r'C:\Users\Aybilge\Desktop\Proje2_SpamTespiti\trainenron/'


Get_confusion_matrix (1) x
[[4810 155]
 [ 108 5037]]
[[4838 127]
 [ 77 5068]]
```


Confusion Matrix

```
Separate_train_and_test.py x Get_confusion_matrix.py x top_10_words.py x Average_word_count.py x
65     return features_matrix, labels
66
67     path = r'C:\Users\Aybilge\Desktop\Proje2_SpamTespiti\trainenron/'
68     dicti = make_Dictionary(path)
69
70     train_matrix, train_labels = extract_features_train(path)
71
72     model1 = MultinomialNB()
73     model2 = LinearSVC()
74     model1.fit(train_matrix, train_labels) #öğrenme işlemi fit fonksiyonuyla
75     model2.fit(train_matrix, train_labels)
76
77     test_dir = r'C:\Users\Aybilge\Desktop\Proje2_SpamTespiti\testenron/'
78     test_matrix, test_labels = extract_features_test(test_dir)
79
80     result1 = model1.predict(test_matrix) #tahmin edilen etiket bilgisi predict
81     result2 = model2.predict(test_matrix)
82     print(confusion_matrix(test_labels, result1))
83     print(confusion_matrix(test_labels, result2))

Get_confusion_matrix (1) x
[[4810 155]
 [ 108 5037]]
[[4838 127]
 [ 77 5068]]
```

Confusion Matrix

 jupyter PROJE2 Last Checkpoint: 04.04.2020 (autosaved)

 Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [6]: import os
from shutil import copyfile #dosyayı hedeflenen yere kopyalar.
import random
import nltk
import os
from nltk.corpus import stopwords #işe yaramayan verileri filtreler. Doğal dil işlemede yararsız kelimeler(veriler) durma
#sözcükleri olarak adlandırılır. Durma sözcüklerini filtrelemiş oluruz
from collections import Counter #nesneleri saymak için kullanılan sınıftır.
import matplotlib.pyplot as plt # Bilmediğiniz bir okuyucuya pyplot modülü olduğunu belirtir.
import numpy as np #Bilimsel hesaplamaları hızlı bir şekilde yapmamızı sağlayan matematik kütüphanesidir.

def make_Dictionary(path):
    emails = [os.path.join(path, f) for f in os.listdir(path)]
    all_words = []
    for email in emails:
        with open(email, encoding='latin1') as m:
            content = m.read()
            all_words += nltk.word_tokenize(content) #boşluk ve noktalama işaretlerine göre ayırır
    dictionary = [word for word in all_words if word not in stopwords.words('english')]
    dictionary = [word.lower() for word in dictionary if word.isalpha()]
    #kelimeleri küçük harfe çeviriyor ve karakterlerin alfabede olup olmadığını kontrol ediyor
    dictionary = Counter(dictionary) #sayıcıyla sözlüğü döndürdük.
    dictionary = dictionary.most_common(10) #en yaygın kullanılan kelimeleri seçip sıklık olarak en az kullanılan kelimeye kadar
    # sözlüğü döndürdük 10 harfin kaç kere tekrar ettiğini bulduk.
    return dictionary

spam_path = r"C:\Users\beril\OneDrive\Masaüstü\Proje2\corpus1\spam"

spam_dict = make_Dictionary(spam_path)
#Oluşturduğumuz grafik için grafiğin apsisine x ordinatına y dedik.
x = [] #apsis
```

Top Ten Words

```

spam_dict = make_Dictionary(spam_path)
#Oluşturacağımız grafik için grafiğin apsisine x ordinatına y dedik.
x = [] #apsis
y = [] #ordinat
my_xticks = []
width = 1/1.5

for i in range(len(spam_dict)): #sözlükteki kelimelerin uzunluğu kadar döndürür.
    x.append(i)
    y.append(spam_dict[i][1])
    my_xticks.append(spam_dict[i][0])

plt.xticks(x, my_xticks)
plt.bar(x, y, width, color="red") #x y ve grafikte bulunacak elemanları belirttik genişliği ve grafiğin rengini belirledik.
plt.show() #grafığı göster

```

Top Ten Words



```

In [9]: import os #python işletim sistemiyle etkileşim için fonksiyonları sağlar.

def word_count(path):
    all_words = []
    emails = [os.path.join(path,f) for f in os.listdir(path)]
    # os.path.join için daha fazla yol bileşenlerini birleştirilmesi için kullanılır.
    #/ ile ayrılmış bileşenleri birleştirmek için kullanabiliriz.
    #os.listdir dizinde var olan her şeyi alacak dosyalar ve dizinler.listdir() yolu ile verilen dizindeki girişlerin adlarını
    #içeren bir liste döndürür.
    for mail in emails:
        with open(mail, encoding="latin1") as m: #Bazı ASCII karakterlerini Latin alfabesine çevirmek için kullandık.
            for line in m:
                words = line.split() #kelimeleri satırlara bölerek buluyoruz.
                all_words += words # yukarıdaki oluşturduğumuz all words listesine kelimeleri ekliyoruz.
    x = len(all_words)
    y = len(os.listdir(path))
    avg_wordcount = x/y #all words listesinin uzunluğu ile listenin yolunun ortalamasını alıyoruz.
    return avg_wordcount

ham_path = r"C:\Users\beril\OneDrive\Masaüstü\Proje2\enron1\ham"
spam_path = r"C:\Users\beril\OneDrive\Masaüstü\Proje2\enron1\spam"

ham_wordCount = word_count(ham_path)
print('Ham postalarının ortalama kelime sayısı:',ham_wordCount)

spam_wordCount = word_count(spam_path)
print('Spam postalarının ortalama kelime sayısı:',spam_wordCount)

Ham postalarının ortalama kelime sayısı: 225.21432461873638
Spam postalarının ortalama kelime sayısı: 234.602

```

Average Words

DATASET:

Enron Dataset 1: Ham Email - 3672 email	Spam Email - 1500 email
Enron Dataset 2: Ham Email - 4361 email	Spam Email - 1490 email
Enron Dataset 3: Ham Email - 4006 email	Spam Email - 1500 email
Enron Dataset 4: Ham Email - 1500 email	Spam Email - 4500 email
Enron Dataset 5: Ham Email - 1500 email	Spam Email - 3675 email
Enron Dataset 6: Ham Email - 1500 email	Spam Email - 4500 email

Toplam Ham Email Sayısı: 16539 mail

Toplam Spam Email Sayısı: 17165 mail

Toplam Email Sayısı: 33704

SONUÇLAR:

Eğitim E-posta Sayısı(Training Set):

33.704 mailin %70'i= 23.594 mail

Test E-posta Sayısı(Test Set):

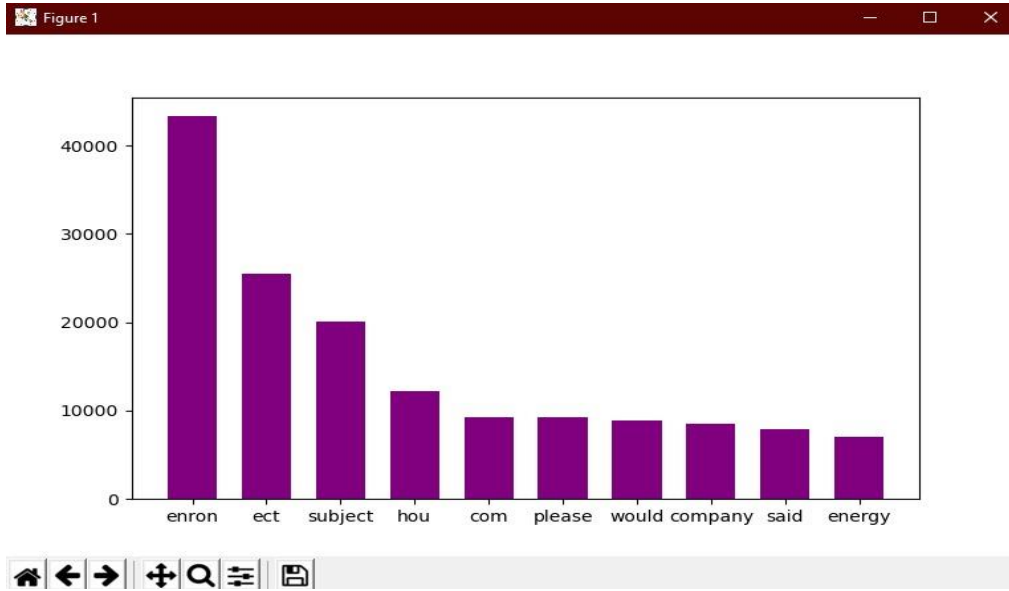
33.704 mailin %30'u= 10.110 mail

Train & Test verisinin sonuçları

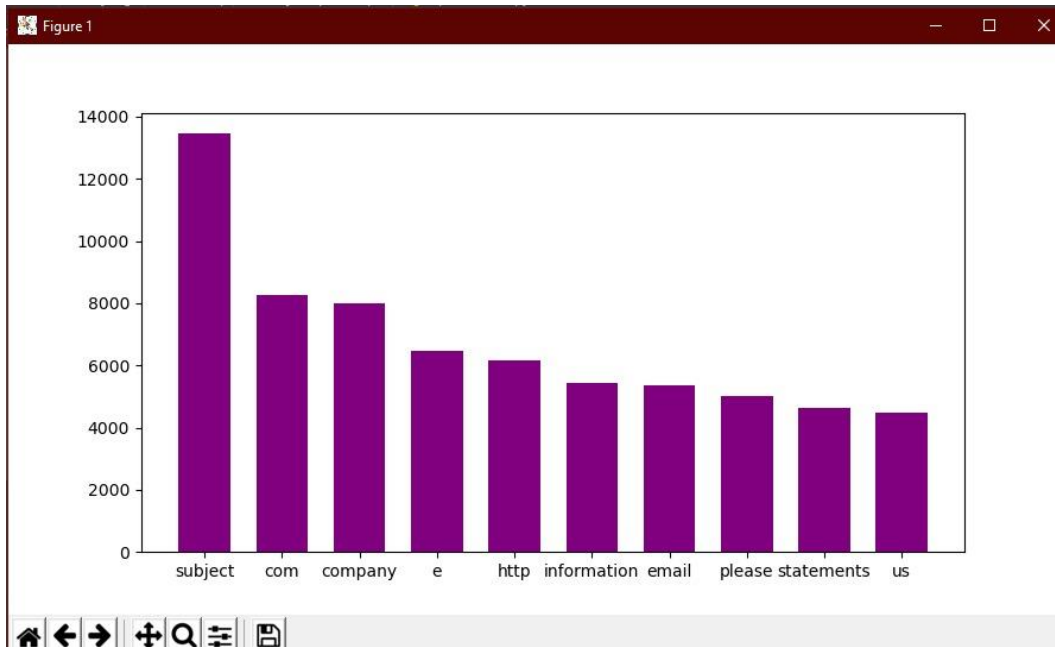
Confusion Matrix:

Multinomial Naive Bayes:	[[4810	155]
	[108	5037]]
Skaler Vektör Makineler:	[[4838	127]
	[77	5068]]

Confusion Matrix sonuçları



Top Ten Words sonuçları-Ham mail



Top Ten Words sonuçları- Spam mail

Ham postaların ortalama kelime sayısı:225.214

Spam postaların ortalama kelime sayısı:234.602

Average Words sonuçları

PROJE METNİ

ÖZET

E-posta en önemli iletişim biçimlerinden biri haline geldi. Her gün birçok mail alıyoruz. Bunların bazıları spam kutumuza düşüyor. Spam, e-posta kullanıcılarına yönelik başlıca tehditlerden biridir. Spam e-postalarındaki bağlantılar, kullanıcıların kötü amaçlı yazılım veya kimlik avı düzenlerine sahip web sitelerine yönlendirilmesine neden olabilir ve bu da alıcının bilgisayar sistemine erişebilir ve bu sistemi bozabilir. Bu siteler aynı zamanda hassas bilgiler de toplayabilir. Bu nedenle, etkili bir spam filtreleme teknolojisi, siber uzayın sürdürülebilirliğine ve toplumumuza önemli bir katkıdır. Spam filtrelemenin birçok yöntemi vardır. Biz bunların arasından içerik tabanlı filtrelemeyi tercih ederek veri setinin üzerinde bir algoritma kurduk.

Anahtar kelimeler;
e-mail, spam, spam filtreleme

GİRİŞ

Elektronik posta (e-posta olarak kısaltılır) İnternet üzerinden iletişim kurmak için hızlı, etkili bir yöntemdir. Gerek kurumlar arası gerekse kişisel iletişimimiz için önemli bir yeri vardır. Dünya çapında 2.3 milyar kullanıcı tarafından e posta kullanımının 2016 yıl sonuna kadar 4.3 milyar sayımına çıkacağı ön görülmektedir. Ancak e-maile artan bağımlılık spam maillerin neden olduğu birçok sorunun ortaya çıkmasına neden olmuştur. Text Retrieval Conference (TREC) göre 'spam' terimi ayırım gözetmeden gönderilen istenmeyen bir maildir.

Spam hacmi yıllar geçtikçe katlanarak arttı ve sadece bir sıkıntı değil, bir güvenlik tehdidi değil bireylere, işyerlerine ve ekonomilere ciddi zararlar vermeye devam ediyor. Spam maillerin atılmasının amacı e-mailin milyonlarca potansiyel müşteriye ulaşmanın çok ucuz bir yolu olması, amatör reklam yönlendiriciler ve doğrudan pazarlamacılar için güçlü bir yoldur. Bu nedenle spam tespiti yöntemleri kullanmalıyız. Böylece hem kendimizi hem de birçok firmayı korumuş oluruz. Spam maillerin gönderilme amaçları ise:

- Ticari reklamlar ve pazarlama faaliyetleri
- Kampanya ve duyurular
- Yanıltıcı ve gerçek olmayan vaatler (Hızlı zengin olma yöntemleri)
- Yasal olmayan veya yarı yasal faaliyet servis duyuruları
- UBE(Unsolicited Bulk E-mail: Talep Edilmemiş Kitlesel E-mail)
- Politik veya ideolojik bir görüşün propagandası
- Belli bir konuda kamuoyu oluşturma
- Kimlik avı saldırısı

E-mail ve spam filtreleri bir e-mail gönderildiğinde, mesaj sistemine girer ve alıcıların posta kutusuna ulaşmaya kadar bir sunucudan diğerine yönlendirilir.

Şu anda spam algılamaya yönelik farklı yaklaşımlar vardır. Bu yaklaşımlar kara liste, toplu e-postaları algılama, mesaj başlıklarını tarama, gri liste ve içerik tabanlı filtreleme içerir:

Kara liste, büyük miktarda spam gönderen IP adreslerini tanımlayan bir tekniktir. Bu IP adresleri Etki Alanı Adı Sistem Tabanlı Kara Delik Listesine eklenir ve listedeki IP adreslerinden gelecek e-postalar reddedilir.

Ancak, spam gönderenler daha fazla sayıda IP adresi kullanarak bu listeleri atlatmaktadır.

Toplu e-postaları tespit etmek, spam'ı filtrelemenin başka bir yoludur. Bu yöntem, bir e-postanın spam olup olmadığını belirlemek için alıcı sayısını kullanır. Ancak, pek çok yasal e-postanın trafik hacmi yüksek olabilir.

İleti başlıklarını taramak, spam'ı tespit etmenin oldukça güvenilir bir yoludur. Spam gönderenler tarafından yazılan program e-postaların başlıklarını oluşturur. Bazen bu başlıklarda standart başlık düzenlemelerine uymamalarına neden olan hatalar vardır. Bu başlıklarda hata olduğunda, e-postanın büyük olasılıkla spam olduğunu gösteren bir işarettir. Ancak, spam gönderenler hatalarından ders alıyor ve bu hataları daha az yapıyor

Gri liste, e-postayı reddetmeyi ve gönderene bir hata mesajı göndermeyi içeren bir yöntemdir. Spam programları bunu görmezden gelir ve e-postayı yeniden göndermez, ancak insanların e-postayı yeniden göndermesi daha olasıdır. Bununla birlikte, bu süreç insanlar için can sıkıcıdır ve ideal bir çözüm değildir.

Geçerli spam teknikleri, etkinliği artırmak için içerik tabanlı spam filtreleme yöntemleriyle eşleştirilebilir. İçerik tabanlı yöntemler, e-postanın spam olup olmadığını belirlemek için e-postanın içeriğini analiz eder. Projemizin amacı, makine öğrenme algoritmalarını analiz etmek ve içerik tabanlı spam filtreleri olarak etkinliğini belirlemektir.

Makine Öğrenmesi (Machine Learning)

Makine öğrenmesi terimi bir makinenin kendi performansını iyileştirme kabiliyetini ifade eder. Bunu, karar vermek için istatistiksel bir model kullanarak ve her yeni denemenin sonucunu bu modele dahil ederek yapar. Temelde, makine deneme yanılma yoluyla öğrenmek için programlanmıştır.

Makine öğrenmesi, yazılım uygulamalarının, açıkça programlanmadan sonuçları tahmin etmede daha doğru olmalarını sağlayan bir algoritma kategorisidir. Makine öğreniminin temel öncülü girdi verilerini alabilen algoritmalar oluşturmak ve yeni veriler mevcut olduğunda çıktıları güncellerken çıktıyı tahmin etmek için istatistiksel analizi kullanmaktır.

Makine Öğrenmesi, bilgisayarların insanlar gibi öğrenmelerini, davranmalarını, özerk bir şekilde gözlemler ve gerçek dünya etkileşimleri şeklinde veri ve bilgi besleyerek öğrenmelerini zaman içinde geliştirmelerini sağlayan bilimdir.”

Makine öğreniminde yer alan süreçler, veri madenciliği ve tahmine dayalı modellemeye benzer. Her ikisi de kalıp aramak ve program eylemlerini buna göre ayarlamak için verilerde arama yapılmasını gerektirir. Birçok insan internette alışverişten makine öğrenmeye ve satın alımlarıyla ilgili reklamlar sunmaya aşinadır. Bunun nedeni, tavsiye motorlarının çevrimiçi reklam dağıtımını neredeyse gerçek zamanlı olarak kişiselleştirmek için makine öğrenmesini kullanmasıdır. Kişiselleştirilmiş pazarlamanın ötesinde, diğer yaygın makine öğrenme kullanım durumları, sahtekarlık tespiti, spam filtreleme, ağ güvenliği tehdidi tespiti, öngörücü bakım ve haber bültenleri oluşturma işlemlerini içerir.

Her gün yüzlerce yayınlanmış birçok farklı makine öğrenme algoritması türü vardır ve bunlar tipik olarak ya öğrenme stiline (yani denetimli öğrenme, denetimsiz öğrenme, yarı denetimli öğrenme) ya da form veya fonksiyondaki benzerliğe göre gruplanır (yani sınıflandırma, regresyon, karar ağacı, kümeleme, derin öğrenme vb. Öğrenme stili veya işlevi ne olursa olsun, makine öğrenme algoritmalarının tüm kombinasyonları aşağıdakilerden oluşur:

- Temsil (bir dizi sınıflandırıcı veya bilgisayarın anladığı dil)
- Değerlendirme (aka hedef / puanlama işlevi)
- Optimizasyon (arama yöntemi; genellikle en yüksek puanlayıcı sınıflandırıcı, örneğin; kullanıma hazır ve özel optimizasyon yöntemleri vardır)

Yapmaya çalıştığınız işe ve türüne bağlı olarak, temel karar ağaçlarının kullanılmasından, yapay sinir ağı katmanlarının kümelenmesine (ikincisi derin öğrenmeye yol açan) öğrenme konusunda farklı yaklaşımlar vardır. Elinizde bulunan veri miktarı ve verinin anlaşılabilir olması çok önemlidir.

Gerçek uygulamalar üzerinde çalışırken yapılan araştırmalar çoğu zaman bu alanda ilerleme sağlar ve sebepler iki yönlüdür: 1. Mevcut yöntemlerin sınırlarını ve sınırlarını keşfetme eğilimi 2. Etki alanı uzmanlarıyla çalışan araştırmacılar ve geliştiriciler ve sistem performansını iyileştirmek için zaman ve uzmanlıktan yararlanma.

Bazen bu aynı zamanda “kaza” ile de ortaya çıkar. Doğruluk geliştirmek için model toplulukları veya birçok öğrenme algoritmasının kombinasyonlarını bir örnek olarak düşünebiliriz.

Öğrenen makineler, insanlar için faydalıdır, çünkü tüm işlem güçleriyle, başkaları tarafından kaçırılmış olabilecek büyük (veya başka) verilerdeki kalıpları daha çabuk vurgulayabilir veya bulabilirler. Makine öğrenimi, insanların sorunları çözme yeteneklerini geliştirmek ve geniş bir sorun yelpazesinde bilinçli çıkarımlar yapmak için kullanılabilir bir araçtır. Dolandırıcıların ve dolandırıcı potansiyeli olan kişilerin tespiti, hastalıkların teşhisine yardımcı olmaktan, küresel iklim değişikliği için çözümler üretmeye kadar doğruya yakın sonuçlar çıkarmamızı sağlayan araçtır.

Makine öğrenmesi bir dizi kurumsal uygulamaya giriyor. Müşteri ilişkileri yönetimi (CRM) sistemleri e-postaları analiz etmek için öğrenme modelleri kullanır ve satış ekibi üyelerinin ilk önce en önemli mesajlara cevap vermelerini ister. Daha gelişmiş sistemler potansiyel olarak etkili tepkiler önerebilir. İş zekası ve analitik satıcıları, kullanıcılara potansiyel olarak önemli veri noktalarını otomatik olarak tanımlamalarına yardımcı olmak için yazılımlarında makine öğrenmesini kullanır. İnsan Kaynakları sistemleri, etkili çalışanların özelliklerini tanımlamak için öğrenme modelleri kullanır ve açık pozisyonlara en iyi adayları bulmak için de kullanılan bir araçtır.

Makine öğrenmesi nasıl çalışır?

Makine öğrenme algoritmaları genellikle denetimli veya denetimsiz olarak kategorize edilir. Denetlenen algoritmalar, algoritma eğitimi sırasındaki tahminlerin doğruluğu hakkında geri bildirim sağlamanın yanı sıra, hem girdi hem de istenen çıktıyı sağlamak için makine öğrenme becerisine sahip bir veri bilimcisi veya veri analisti gerektirir. Veri bilimcileri, modelin tahminleri geliştirmek için hangi değişkenleri veya özellikleri analiz edip kullanması gerektiğini belirler. Eğitim tamamlandığında, algoritma öğrenilenleri yeni verilere uygulayacaktır.

Denetimsiz algoritmaların istenen sonuç verileriyle eğitilmesi gerekmez. Bunun yerine, verileri incelemek ve sonuçlara varmak için derin öğrenme denilen yinelemeli bir yaklaşım kullanıyorlar. Denetimsiz öğrenme algoritmaları- sinir ağıları olarak da adlandırılır- görüntü tanıma, metinden konuşmaya ve doğal dil oluşturma da dahil olmak üzere denetimli öğrenme sistemlerinden daha karmaşık işleme görevleri için de kullanılır.

Bu sinir ağırları milyonlarca eğitim verisi örneğini birleştirerek ve birçok değişken arasındaki sık sık ince bağıntıları otomatik olarak tanımlayarak çalışır. Bir kez eğitildikten sonra, algoritma yeni verileri yorumlamak için kendi dernek bankasını kullanabilir. Bu algoritmalar, büyük miktarda eğitim verisi gerektirdiğinden, yalnızca büyük veriler çağında uygulanabilir hale gelmiştir.

Veri Madenciliği

Genel terim olarak neyin keşfedileceğine dair ön bilgiye sahip olmadan büyük veri dosyalarından birtakım teknikler kullanılarak gerekli bilginin elde edilme sürecidir. Bu yararlı bilgiler genellikle veri içerisindeki önceden bilinmeyen ve hatta beklenmeyen ilişkilerden, örneklerden oluşur. Kullanılan teknikler makine öğrenmesi ve modelleme olarak adlandırılır.

Genel bir yanlış veri madenciliği ile çok büyük miktardaki verinin yalnızca çok üstün teknolojilerden geçirilerek örneklerin bulunması ve iş problemlerine sihirli çözümlerin getirildiği düşüncesidir. Veri madenciliği yöntemi geleneksel istatistiksel uygulamalara göre daha otomatik olmasına rağmen bu düşünce doğru değildir. Kullanılan teknikler makine öğrenmesi ve modelleme olarak adlandırılır. Kullanıldığı alanlar;

- Telefonlardaki, kredi kartlarındaki dolandırıcılıkların ortaya çıkması için model geliştirilmesi
- İyi veya kötü satış beklentilerinin ortaya çıkması
- Bir web sitesinin incelenecek bir sonraki sayfanın tahmini
- Poliçelerini, hesaplarını, aboneliklerini iptal etmeye eğilimli müşterilerin belirlenmesi
- Farklı örneklerdeki gruplarına göre müşterilerin sınıflandırılması
- Satılan ürünle birlikte diğer ürünlerin belirlenmesi
- Bir üretim sürecini etkileyen önde gelen faktörlerin belirlenmesini sağlar.

Herhangi bir veri madenciliği projesinde hangi spesifik veri madenciliği tekniğinin uygun olacağını düşünmeden önce iş problemlerinin ve verilerinin değerlendirilmesi gerekir

- Analiz edilecek veri kullanmaya elverişli mi?
- Analiz edilecek veri ile ilgili tüm faktörleri içeriyor mu?
- Analiz edilecek veri çok kirli mi?
- Analiz edilecek veri yeterli büyüklükte mi?
- Analiz edilecek veri hakkında gerekli bilgi var mı?

Veri Madenciliği çalışmaları birkaç aşamadan oluşan bir süreçtir. Bu aşamalardan biri hatta en önemlisi olan modelleme öncesi verinin hazırlanması, proje sürecinin belki de en uzun ve en çok emek gerektiren bölümüdür.

Veri temizlemeden veya hazırlanmadan kurulan model ne kadar iyi olursa olsun başarılı olma ihtimali oldukça düşüktür.

Veri madenciliğinde kullanılan verinin kalitesi ve niteliği ne kadar artarsa elde edilecek tahminlerin veya sonuçların geçerliliği de o kadar artacaktır.

Boş veya kayıp değerlerin yönetiminin yanı sıra veri içerisindeki anormalliklerin belirlenmesinde kullanılır.

Veri okunduktan ve gerekli olan tüm veri kaynakları birleştirildikten sonra veri temizleme işlemlerinin ilk adımı verinin genel kalitesini değerlendirmektir.

Verinin genel durumu değerlendirildikten sonra veri kalitesini artırıcı yöntemler uygulanır.

Veri seti kayıp değerlerden ve limitlerin dışında kalan gözlemlerden temizlenmiş durumdadır.

Gerekli olduğu takdirde Distinct işlemcisi ile tekrar eden gözlemlerin de kaldırılmasını sağlayabilir. CRISP-DM 6 aşamadan oluşur.

Business Understanding (İş Anlama)

En önemli aşamadır. Projenin amaç ve gereksinimlerinin iş perspektifi ile anlaşılması

- İş amaçlarının ve başarı kriterlerinin tanımlanması
- Durum değerlendirmesinin yapılması
- Projenin amaçlarının belirlenmesi
- Proje planının oluşturulması

Verinin Hazırlanması (Data Preparation)

- Veri ambarından verinin çekilmesi
- Veri tabanı içerisindeki ya da IBM SPSS tabloların birbirine bağlanması
- Farklı sistemlerdeki veri dosyalarının birleştirilmesi
- Tutarlı olmayan değişken değerlerinin tutarlı hale gelmesi
- Kayıp, yanlış girilmiş ya da aykırı değerlerin tanımlanması
- Veri seçimi
- İlgili değişkenlerin dönüştürülmesi

Modelleme (Modelling)

Analiz yöntemleri veriden gerekli bilgiyi çıkarmak için kullanılır. Bu aşama model tekniklerinin seçilmesi test dizaynının üretilmesi, modelin oluşturulması ve değerlendirilmesi

Değerlendirme (Evaluation)

Modeli uygulamadan önce, modelin ayrıntılarıyla değerlendirilmesi ve oluşturulan modelin çalıştırılması yeniden incelenmesi aşamasıdır.

Uygulama (Deployment)

Eğer bir modelin amacı veri bilgisinin artırılması ise, kazanılan bilgi düzenlenmeli ve karar vermede organizasyonun kullanılacağı şekilde bir yol sunulmalıdır.

Veri madenciliğindeki başarısızlık nedenleri

- Kötü veri
- Organizasyondaki dirençler

(Yararlı sonuçları hakkında ileri düzeyde eğitim verilmesi, organizasyonun sadece belirli bir bölümü üzerinde yürürlüğe konulmasıdır.)

- Uygulanabilir sonuçların olmaması

(Karar vermede yasal olmayan faktörlerin bulunmasıdır.)

- Sebep ve etki problemleri

(Modeldeki tahmin edicilerin hedef değişkenlerden önce meydana geldiğine emin olmamız şarttır.)

Veri madenciliği için gerekli yetenekler

- İşin anlaşılması
- Veritabanı bilgisi

Veri madenciliği yöntemleri

- Uygulama
- Takım
- Eğitim planı

Veri Madenciliği yöntemleri ikiye ayrılır.

Tahmini Yöntemler (Sınıflandırma)

- Karar Ağaçları
- Yapay Sinir Ağları
- Time Series Analitics
- Karar Destek Makinesi
- Regresyon

Tanımlayıcı Yöntemler

- Kümeleme
- Birliktelik Analizi
- Özetleme
- Terminal İstatistik
- İstisna Analizi

Regresyon

“Regresyon çözümlemesi, bir bağımlı değişkenin başka bağımsız değişkenlere olan bağımlılığını, bağımlı değişkenin ana kütle ortalama değerini, bağımsız değişkenin yinelenen örneklerdeki bilinen ya da değişmeyen değerleri cinsinden tahmin etme ve/veya kestirme amacı ile inceler.”

- Doğrusal Regresyon

Doğrusal regresyon modeli, iki ya da daha fazla değişken arasındaki doğrusal ilişkiyi açıklar. Açıklanan değişkene bağımlı değişken, açıklayıcı değişkenlere ise bağımsız değişken adı verilir. Örneğin, gelir düzeyi ve eğitim düzeyi arasındaki ilişkiyi, öğrencilerin devamsızlık yaptığı günler ile başarıları arasındaki ilişkiyi açıklamak için regresyon modeli kullanılabilir.

- Lojistik Regresyon

Lojistik Regresyonda sembolik bir hedef değişkenin tahmininde kullanılır. Gözlemlerin hedef değişkenin hangi kategorisine ait olduğunu tahmin eden bir olasılık fonksiyonudur. S şeklindeki eğri lojistik eğridir ve bu nedenle tekniğin adıdır. Olasılık ölçüldüğü için hedef değişken değeri 0 ile 1 arasında değişir.

Lojistik Regresyon yapmanın iki genel amacı vardır.

- Olasılık üzerinde bireysel değişkenlerin etkisine ek olarak değişken kümelerinin etkisinin belirlenmesi
- Verilerin tahmin edici kümesinden en yüksek tahmin doğruluğuna ulaşmak

Lojistik regresyonda öncelikle uygun bir takım tahmin ediciler seçilmelidir ve öncelikle veride normal olmayan örüntülerin belirlenmesi, aykırı değerler, kayıp veri problemleri gibi sorunlar incelenmelidir. Denklemin tahmin edilmesi ve değişkenlerin teker teker etkilerinin incelenmesinden sonra verinin lojistik regresyon varsayımlarını karşılayıp karşılamadığını kontrol etmek gerekir.

Lojistik regresyonda;

- Bağımsız değişkenler sürekli ve kategorik olabilir.
- Denklemden ilgili tüm tahmin edicilerin bulunması ve ilişkinin formu doğrusal olmalıdır.
- Hata ortalaması 0 olmalıdır.
- Hata ve bağımsız değişkenler arasında ilişki olmamalıdır.
- Bağımsız değişkenler arasında çoklu bağlantı sorunun olmamalıdır.

Karar Ağaçları

“Karar ağaçları, tek bağımlı değişken ve çok sayıda bağımsız değişkene sahip olmaları açısından regresyon modellerine benzerler. Bununla birlikte, ek olarak, veriden regresyon modellerine alternatif olabilecek farklı ve kullanışlı örüntüler keşfederler.” Karar ağaçları, bağımlı değişkenin kategorik olduğu durumlarda lojistik regresyona alternatif oluşturabilecek bir yöntemdir. Kolayca kural cümleciklerine çevrilebilir olmaları, sürekli ya da kesikli veriler ile çalışabilmeleri, eksik veya hatalı veriler ile tahminleme yapabiliyor olmaları karar ağaçlarının avantajlarından. Ayrıca parametrik olmayan yöntemler arasındadır. Bu, karar ağaçlarının uzay dağılımı veya sınıflayıcı yapısı ile ilgili varsayımlara uymak zorunda olmadığı anlamına gelir. Bununla birlikte, eksik veya hatalı verilere duyarsız olması ve yaprak düğümlerde sıkıntı içermesi de dezavantajı olabilmektedir.

Bilinen en popüler karar ağacı algoritmaları C&RT, CHAID tir.

Chaid Algoritması

Chaid algoritması, kategorik bağımsız değişkenler ile çalışmayı tercih ettiğinden, modele giren bağımsız değişkenleri, sürekli olmaları halinde bölerek kategorik hale getirir. Bağımsız değişken çok fazla kategoriye sahip ise, bu durumda kategori sayısını indirgeyerek ağacı basitleştirme yoluna gider.

C&RT (Sınıflandırma ve Regresyon Ağacı)

C&RT algoritmaları, bağımlı değişkenin kategorik olduğu durumlarda sınıflandırma, sürekli olduğu durumlarda tahminleme modeli kuran bir karar ağacı algoritmasıdır. C&RT algoritmaları için birincil amaç, mümkün olan en iyi doğruluğu olan modeli kurabilmektir. En iyi doğruluk ise minimum maliyetli tahminler yapılmasını içerir. Minimum maliyetli tahminler yapılması, en düşük yanlış tahmin oranına yani yanlış sınıflandırılan verinin az olmasına sahip olunması demektir.

Sinir Ağları (Neural Network)

Öngörüsül model oluşturmak için kullanılan sinir ağlarında girdi katmanı hedef değişkeninin tahmin edilmesinde kullanılacak bütün değişkenleri içerir. Çıktı katmanında tahminin hedefini içeren bir çıktı değişkeni vardır. Girdi ve çıktı değişkenleri sayısal ya da sembolik tipte olabilir. Saklı tabakada ise hedef değişkenlerin bir önceki katmanda birleştiği çok sayıda nöron içerir. Bir ağ çok sayıda saklı katman içerebilir fakat bu sayının olabildiğince az tutulmasında yarar vardır. Sinir ağlarının bir katmanındaki nöronlar sonraki gelen katmandaki nöronlarla bağlıdır.

Sinir ağlarının veri ve sonuçlar arasındaki ilişkiyi öğrenmesi eğitim olarak da adlandırılabilir.

Bir eğitim sonunda bu ağ edindiği deneyim ile henüz görülmemiş yani yeni bir veri ile karşılaştığında bu veri hakkında bir karar verebilir ya da tahminde bulunabilir.

İki çeşit sinir ağı algoritması vardır. Bunlar Multi-Layer Perceptron (MLP) ve Radial Basis Function Network (RBFN)'dir.

Naive Bayes Classifier

Naive Bayes sınıflandırıcısının temeli Bayes teoremine dayanır. Lazy (tembel) bir öğrenme algoritmasıdır aynı zamanda dengesiz veri kümelerinde de çalışabilir. Algoritmanın çalışma şekli bir eleman için her durumun olasılığını hesaplar ve olasılık değeri en yüksek olana göre sınıflandırır. Az bir eğitim verisiyle çok başarılı işler çıkartabilir. Test kümesindeki bir değerın eğitim kümesinde gözlemlenemeyen bir değeri varsa olasılık değeri olarak 0 verir yani tahmin yapamaz. Bu durum genellikle Zero Frequency (Sıfır Frekans) adıyla bilinir. Bu durumu çözmek için düzeltme teknikleri kullanılabilir. En basit düzeltme tekniklerinden biri Laplace tahmini olarak bilinir. Kullanım alanlarına örnek olarak gerçek zamanlı tahmin, çok sınıflı tahmin, metin sınıflandırması, spam filtreleme, duyarlılık analizi ve öneri sistemleri verilebilir.

SONUÇ

Projemiz spam detection(tespiti) idi. Bu vesile ile birçok yerde veri aradık. Veri seti aramamızın sebebi öncelikle veriyi CRISP-DM aşamalarından geçirmekti. Birçok yerden az az bulduğumuz verileri toplayıp bir veri seti oluşturduk. İlk aşama Business Understanding(iş anlama) işin amaçlarının başarı kriterlerinin belirlenmesi, durum değerlendirmesi yapılması projenin amaçlarının belirlenmesi ve proje planının oluşturulması . Daha sonra Data Preparation(Verinin hazırlanması) birçok veri setini birleştirip bir veri seti oluşturduk. Veri setinin hazırlanması, temizlenmesi, tutarsız değişken değerin tutarlı hale gelmesini sağlamıştık. Spam tespiti yapabilmek için spam algılamaya yönelik 4tane farklı yaklaşım var. Kara liste, toplu e-postaları algılama, mesaj başlıklarını tarama, gri liste ve içerik tabanlı filtreleme gibi. Bizim veri setimize göre en uygun olanı içerik tabanlı filtreleme yapmakti dolayısıyla biz kelime sayısından yola çıkarak bu projeye başlangıcımızı yaptık.

Sınıflandırma yapabilmek için verinin train(eğitim) ve test verisi olarak ayrılması gerekiyordu. Veri setinin %70ini eğitim veri seti olarak ve veri setinin %30unu test veri seti olarak aldık. Maillerin %70 i train veri setine girer ve train veri setinde olmayan tüm mailler test veri setine girer. 70 'e 30 olmasının sebebi kategori katsayısını değiştirerek veri dengelenmesini sağlamaktır. Veri setinde gözlem sayısını artırma (boost işlemi) ya da gözlem sayısı çok olanların gözlem sayısını düşürme (reduce işlemi) yapılır. Dosya işlemlerini yaparak veri setimizi train ve test olarak ayırmış bulunmaktayız. Algoritma ile eğitilecek verilerimizi ayrıştırdık. Dosya içerisindeki dosyaların sayısını gözlemledik ve train test dosyalarımız ayrıldı.

Makine öğrenmesinde bizim için en uygun olacak kütüphaneyi seçtik. Scikit-learn; sınıflandırma, regresyon ve kümeleme için pek çok öğrenme algoritmasına sahip. Daha sonraki adımımızda hangi algoritmayı veya karar ağacını kullanabilirizi araştırdık ve genelde sınıflandırma(classifier) algoritması olan Naive Bayes spam tespitinde daha çok kullanılan bir algoritma olduğunu belirledik. Bu algoritmanın çalışma şekli bir eleman için her durumun olasılığını hesaplıyor ve olasılık değeri en yüksek olana göre sınıflandırıyor. Az bir eğitim verisiyle çok başarılı işler çıkartabilir. Test kümesindeki bir değerın eğitim kümesinde gözlemlenemeyen bir değeri varsa olasılık değeri olarak 0 verir yani tahmin yapamaz.

Bunun yanında matematiksel işlemler için Numpy ve sınıflandırma algoritması için Navie Bayes'i projemize import ettik. Sklearn kütüphanesiyle beraber kullandığımız Navie Bayes'i çok terimli model sınıflandırması için seçtik.

Projemizin bu bölümünde sonuç olarak matris değerler görmek istiyoruz. Bunun için ilk önce 3000 kelimenin seçildiği bir sözlük oluşturduk. Bu sözlük makinemize öğreteceğimiz belli kelimelerden oluşuyor. Bu kelimeleri en çok kullanılanlardan en az kullanılanlara göre seçtik. Bu işlem hem makinemizin öğrenmesini kolaylaştıracak hem de bizim hangi kelimelerin daha yaygın olarak kullanıldığını matematiksel olarak görmemizi sağlayacak. Bir önceki hafta elimizdeki verileri %70 eğitim ve %30 test verisi olacak şekilde düzenlemiştik.

Projemizin bu kısmında eğitim veri setimiz için bir algoritma kurduk. Train(eğitim) veri setimizin içinden çektiğimiz kelimelerle bir liste oluşturduk. Özelliklerin olduğu matrisin ve etiketlerin olduğu matrisin tek bir matrisini oluşturduk. Özellik matrisinin içinde sözlükteki satırları, email sayısını ve sütun sayısını veriyoruz. Etiket matrisinin içinde ise kaç tane mailin spam, kaç tane mailin ham olduğunu hesaplıyoruz. Oluşturulan matrisi makinemize belirli fonksiyonlarla öğreteceğiz.

Test veri setimiz için bir algoritma kurduk. Test veri setimizin içinden çektiğimiz kelimelerle bir liste oluşturduk. Train veri setimizde olduğu gibi özelliklerin olduğu matrisin ve etiketlerin olduğu matrisin tek bir matrisini oluşturduk. Özellik matrisinin içinde sözlükteki satırları, email sayısını ve sütun sayısını veriyoruz. Etiket matrisinin içinde ise kaç tane mailin spam, kaç tane mailin ham olduğunu hesaplıyoruz. MultinomialNB ve LinearSVC ile iki model oluşturuyoruz. Bu sayede iki model üzerinden programımızı test edebilmiş oluyoruz. MultinomialNB; Naive Bayes'in bir çeşididir. Belge sınıflandırma problemi için kullanılır. Bir belgenin spor, teknoloji vb. kategorisine ait olup olmadığını bulmak için kelime sıklığına göre sınıflandırır. LinearSVC; amaç verilerimizi bölen veya kategorize eden "en uygun" bir hiperplane döndürerek sağladığımız verilere uymaktır. Oluşturduğumuz bu modelleri fit() fonksiyonuyla makinemize öğretiyoruz. Makineye öğretme işlemi verilerin büyüklüğüne göre uzunluk ve kısalık gösteriyor. Biz bu işlem için 2-2.5 saat bekledik. Makinemiz öğrenme işlemini tamamladıktan sonra predict() fonksiyonuyla tahmin ettiğimiz etiketlerin bilgilerini çekiyoruz. Her iki modelimiz için matrislerimizi yazdırıyoruz.

Spam filtrelemede izleyeceğimiz yol olan kelime filtreleme ile devam ettik ve en çok hangi kelimeler kullanılıyor tespitini yapmaya çalıştık. Bunu yapmamızın sebebi spam maillerde aynı kelimelerin çokça kullanılıyor olması. Spam maillerde en çok kullanılan 10 kelimeyi grafikte göstermek istedik. Grafiği matplotlib kütüphanesi kullanarak yaptık. Bunun yanısıra matematiksel işlem yapacağımız için numpy kütüphanesini de import ettik. Daha sonra en çok kullanılan 10 kelimeyi listeledik

Sonuç olarak ham maillerde en çok kullanılan kelimeler: enron, ect, subject, hou, com, please, would, company, said, energy, spam maillerde en çok kullanılan kelimeler: subject, com, company, e, http, information, email, please, statements,us olarak belirlendi ve tüm sayısal sonuçlarımız tablolar kısmında belirtilmiştir.

KAYNAKÇA

- <https://towardsdatascience.com/spam-detection-with-logistic-regression-23e3709e522>
- <https://ese.wustl.edu/ContentFiles/Research/UndergraduateResearch/CompletedProjects/WebPages/sp14/SongSteimle/WebPage/motivation.html>
- <https://web.stanford.edu/class/cs124/lec/naivebayes.pdf>
- https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering
- <https://medium.com/secure-and-private-ai-math-blogging-competition/spam-detection-and-filtering-with-naive-bayes-algorithm-f6c2ac181174>
- <https://www.kdnuggets.com/2017/03/email-spam-filtering-an-implementation-with-python-and-scikit-learn.html>
- <https://www.mailchannels.com/what-is-spam-filtering/>

EK:

Proje yazım kuralları

No	Proje düzeni için sorular (X işareti koyunuz)	Evet	Hayır
1	Projenizin kapağı var mı?	X	
2	Özet yazıldı mı?	X	
3	İçindekiler kısmı var mı?	X	
4	Tablolar listesi var mı?	X	
5	Şekiller listesi var mı?	X	
6	Projeniz Times New Roman formatında mı?	X	
7	Proje, 12 punto olarak mı yazıldı?	X	
8	Sayfa düzeni, sol ve üst 4 cm ile sağ ve alt 2,5 cm mi?	X	
9	1. derece başlıklar; kalın (bold), sola yaslı ve tüm harfleri büyük mü?	X	
10	2. derece başlıklar; kalın, sola yaslı ve baş harfleri büyük mü?	X	
11	3. derece başlıklar; kalın, sola yaslı ve sadece ilk baş harfi büyük mü?	X	
12	1. Derece başlıklar yeni sayfadan başladı mı?	X	
13	Başlıklardan sonra ":" olmayacak.	X	
14	Projede, paragraf arası 1,5 satır aralığı mı?	X	
15	Paragraf arası boşluk bırakılmayacak.	X	
16	Paragraf başlarında girinti olmayacak.	X	
17	Metin kısmı iki tarafa yaslı olarak yazıldı mı?	X	
18	Sayfa numarası, sayfa altında ortaya verildi mi?	X	
19	Her tablonun numarası ve adı var mı?	X	
20	Tablo numarası Tablo 1, Tablo 2.şeklinde gidiyor mu?	X	
21	Tablonun adı, tablonun üstüne yazıldı mı?	X	
22	Metin içinde her tablo açıklandı mı?	X	
23	Her şeklin numarası ve adı var mı?	X	
24	Şekilin numarası Şekil 1, Şekil 2, şeklinde gidiyor mu?	X	
25	Şekilin adı, şeklin altına yazıldı mı?	X	
26	Metin içinde her şekil açıklandı mı?	X	
27	Tablo sola yaslandı mı?	X	
28	Şekil sayfaya ortalandı mı?	X	
29	İmla kurallarına dikkat edildi mi?	X	
30	Her kaynak metin içinde gösterildi mi?	X	
31	Kaynakça kısmında kaynaklar formatta belirtildiği gibi gösterildi mi?	X	
32	Kaynakça istenilen sıraya göre düzenlendi mi?	X	

Çalışmamızı hepsini EVET olacak şekilde düzenlediğimizi, kontrol ettiğimizi taahhüd ederiz.	İmza
Adı Soyadı-NO: BERİL DİNDAR	B.Dindar
Adı Soyadı-NO: AYŞE AYBİLGE MURAT	A.A.Murat