|  |  |  |
| --- | --- | --- |
|  |  |  |

**T.C.**

**CUMHURİYET ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

|  |  |  |
| --- | --- | --- |
|  | GitHub Projelerinde Sürüm Tahmini  **2017141009**  **Hatice Berfin ÇINAR**  **LİSANS BİTİRME PROJESİ** |  |

**12-2020**

**SİVAS**

**TEZ BİLDİRİMİ**

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Hatice Berfin ÇINAR

Tarih:16.12.2020

**ÖZET**

**Github Projelerinde Sürüm Tahmini**

**Hatice Berfin ÇINAR**

**Danışman: Arş.Gör.Abdulkadir ŞEKER**

Bu çalışmanın amacı ilerleyen bir projede yapılan commitleri analiz ederken, yapılması gereken minör veya majör sürümlerin tahmin edilmesini sağlamak.

**Anahtar Kelimeler:**

* Commit
* Dağıtık Programlama
* Açık Kaynak Programlama
* Kod versiyonlama
* Releases
* Pull Request

İÇİNDEKİLER

[ÖZET iv](#_Toc62741904)

[İÇİNDEKİLER v](#_Toc62741905)

[1. GİRİŞ 5](#_Toc62741906)

[1.1. Projedeki Kişiler ve Görevleri 5](#_Toc62741907)

[2. KAYNAK ARAŞTIRMASI 6](#_Toc62741908)

[2.1. Ön Araştırma için Kaynak Araştırması 6](#_Toc62741909)

[2.2. Github’ta En Çok Tercih Edilen Programlama Dilleri 6](#_Toc62741910)

[2.2.1. Snowball Stemmer](#_Toc62741911) 6

2.2.2. Porter2 Stemmer………………..………………………………......................6

2.2.3. Lancaster Stemmer……………………………………………………………6

2.2.4. Poster Stemmer………………………………………………………………..6

[3. MATERYAL VE YÖNTEM 7](#_Toc62741912)

[3.1. Ön Araştırma Konuları 7](#_Toc62741913)

[3.1.1. Dağıtık Kodlama 7](#_Toc62741914)

[3.1.2. Açık Kaynak Kodlama 7](#_Toc62741915)

[3.1.3. Kod Versiyonlama 7](#_Toc62741916)

[3.1.4. GitHub Repository 8](#_Toc62741917)

[3.1.5. GitHub Contributors 8](#_Toc62741918)

[3.1.6. GitHub Issue 8](#_Toc62741919)

[3.1.7. GitHub Commit 9](#_Toc62741920)

[3.1.8. GitHub Pull Request 9](#_Toc62741921)

[3.1.9. GitHub Releases 10](#_Toc62741922)

[3.2. Release Notlarının Araştırılması](#_Toc62741923) 13

[3.2.1. Mozilla Firefox Release Notları 13](#_Toc62741924)

[3.2.2. Mozilla Thunderbird Release Notları 15](#_Toc62741925)

[3.2.3. OBS Project Release Notları 17](#_Toc62741925)

[3.2.4. Slack Release Notları 18](#_Toc62741925)

[3.3. Text Pre Processing İşlemleri 20](#_Toc62741923)

[3.3.1. Text Pre Processing 20](#_Toc62741925)

[3.3.2. Text Hero ile Text Pre Processing 23](#_Toc62741925)

[3.3.3. Stemming](#_Toc62741925) 25

[3.3.4. Stemming Kütüphaneleri Karşılaştırılması 26](#_Toc62741925)

[3.3.5. Porter2 Stemmer ile Veri Setinin Temizlenmesi 27](#_Toc62741925)

[3.4. Makine Öğrenmesi 30](#_Toc62741923)

3.4.1 .Makine Öğrenmesi Temel Kavramlar Supervised Learning……........................30

3.4.1.1.Feature ve Label Nedir……………………………………………………..….....30

3.4.1.2.Model Nedir………………………………………………………………..…......31

3.4.1.3.Regresyon ve Sınıflandırma……………………………………………………...31

3.4.1.4.Lineer Regresyon………………………………………………………….……...31

3.4.1.5.Training ve Loss………………………………………………………………….33

3.4.1.6.Train ve Test……………………………………………………………….…......33

3.4.1.7.Lojistik Regresyon………………………………………………………………..33

[3.4.2. Makine Öğrenimi Algoritmaları ve Karşılaştırmalar……………………………........34](#_Toc62741925)

3.4.2.1. Lineer Regresyon…………………………………………………………….……..34

3.4.2.2.Lojistik Regresyon……………………………………………………………..……35

3.4.2.3***.***Doğrusal Regresyon K-En Yakın Komşular (KNN)……………………….……37

3.4.2.4.Karar Ağacı (Decision Tree)…………………………………………………...……38

3.4.2.5.Destek Vektör Makinesi(SVM)……………………………………………..………39

3.4.2.6.Rastgele Orman(Random Forest………………………………………………….....40

3.4.2.7.Naive Bayes…………………………………………………………………….…...41

[3.4.3. Müşteri Yorumları Hissiyat Analizi Uygulamas………………………………….…..41](#_Toc62741925)

3.4.3.1Veri Seti Ön İşleme Adımları ………………………………………………….…....41

3.4.3.2 Uygulama Bölümü ……………………………………………………………….……42

3.4.4 Relase Notları Veri Seti üzerinde Makine Öğrenmesi ve Derin Öğrenme Algoritmalarınının Uygulanması …………………………………………………………....44

3.4.4.1.Naive Bayes Algoritmasının Uygulanması………………………………………......44

3.4.4.2SVM Algoritmasının Uygulanması ………………………………………………......45

3.4.4.3.LSTM Algoritmasının Uygulanması………………………………………………….46

3.4.4.4.CNN Algoritmasının Uygulanması………………………………………………......47

[4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA………………………………………..](#_Toc62741926)49

[4.1. Projede Kullanılan Yöntem Araştırması……………………………………………..](#_Toc62741927)49

[5. SONUÇLAR VE ÖNERİLER………………………………………………………….](#_Toc62741928)49

[5.1 Sonuçlar……………………………………………………………………………](#_Toc62741929)49

[KAYNAKLAR………………………………………………………………………50](#_Toc62741930)

**1. GİRİŞ**

Amaç ilerleyen bir projede yapılan commitleri analiz ederken, yapılması gereken minör veya majör sürümlerin tahmin edilmesini sağlamak.

## **1.1. Projedeki Kişiler ve Görevleri**

Projede 3 farklı ekip ve her ekipte ikişer kişi olmak üzere toplamda 6 öğrenci vardır.

1. Ekip : 3 aktif projeye ait release (sürüm) notları tespit eder.
   1. Son 3 yıl en çok kullanılan dil veya konusunu bulur.
   2. Karar verilen konsept üzerinden, en çok commit veya katkı olan projeler bulunur.
   3. Projelere ait release notları bulunur.

Ekip Üyeleri : Doğukan DURAK – Asiye Nur İLERİ

1. Ekip : Bu notlarda geçen kelimeler analiz edilerek minör majör ayrımı yapar.
   1. Kelimeleri NLTK dan geçir, sadece gerekli olanlar elde eder.
   2. Kök bulma, ekleri silme, stopword kelimeleri kaldırır.
   3. Bir veri kümesi oluşturur. Kelimeler ve minör / majör olma durumunu ayırt eder.

Ekip Üyeleri : Hatice Berfin Çınar – Tolga TANRISEVER

1. Ekip: Derin öğrenme tabanlı bir uygulama ile yeni bir proje üzerinde sürüm tahmini yapar.
   1. Makine öğrenmesi veya Derin öğrenme algoritmalarını öğrenir.
   2. Doğal dil işlemede sınıflandırma problemi için bir uygulama yapar.
      1. En az 3 farklı yöntem önerilir.
   3. Bu uygulamayı önceki bölümden gelen veri kümesi üzerinde test eder.

Ekip Üyeleri : İsmail Can ARDAHAN - Muhammet Fatih ŞAHİN

**2.KAYNAK ARAŞTIRMASI**

## **2.1. Ön Araştırma için Kaynak Araştırması**

Verilen ön araştırma konuları için aşağıdaki belirtilen konular belirli internet sitelerinden araştırıldı;

* Dağıtık Kodlama
* Açık Kaynak Kodlama
* Kod Versiyonlama
* GitHub
* GitHub Repository
* GitHub Contributor
* GitHub Issue
* GitHub Commit
* GitHub Pull Request

## **2.2. Doğal Dil İşleme’de Kullanılan Kütüphaneler (Stemmer)**

2.2.1) Snowball Stemmer

2.2.2) Porter2 Stemmer

2.2.3) Lancaster Stemmer

2.2.4) Porter Stemmer

**3. METERYAL VE YÖNTEM**

**3.1. Ön Araştırma Konuları**

**3.1.1Dağıtık Kodlama (Sistem) Nedir?**

Dağıtık kodlama (sistem), birden fazla bilgisayar veya sunucunun birbirleriyle iletişim kurması ve bir ağ bütünü olarak çalışmasına denir.

Dağıtık kodlama bir ağ ile kurulmuştur.

Yazılımla, ağdaki bileşenler arası uyumluluk ve saydamlık sağlanır.

Bu büyük ağdaki donanımlar(bilgisayarlar, sunucular) kullanıcıya tek bir bilgisayar gibi davranırlar ve en iyi performansı amaçlar.

Bu kaynaklar donanımsal bileşenler (hdd, yazıcı) olabileceği gibi dosyalar, veri tabanı vb. servislerdir.

**3.1.2.Açık Kaynak (open source ) Kod Nedir?**

Her yazılımcının ulaşabileceği, üzerinde oynayabileceği, istediği gibi değiştirebileceği kodlara açık kaynak kod denir.   
Açık kaynak kodlar sayesinde herkes yapacağı projeyi rahatlıkla değiştirebilir ve geliştirebilir.

**3.1.3Kod Versiyonlama Nedir?**

Kod versiyonlama diğer adıyla “Versiyon Kontrol Sistemleri (VCS)” dir.

Versiyon veya sürüm kontrolü ise, bir ya da daha fazla dosya üzerinde yapılan değişikliklerin kayıt ve takip edilmesini ve gerekli bir durumda da belirli işlem noktalarına geri dönülebilmesini mümkün kılan bir sistemdir.

İlk versiyon kontrol sistemi, SCCS yani Source Code Control System’dir.

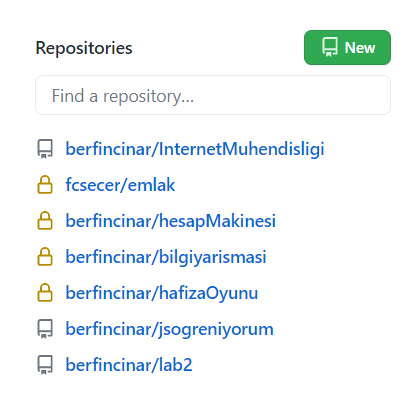
Daha sonra Özgür Yazılım Vakfı tarafından 1982 yılında GNU RCS yani Revision Control System yayınlanmıştır. İlk nesil versiyon kontrol sistemleri o tarihlerde başlamıştır. İkinci nesil versiyonlar da 1990’lar da SVN ile başlamış ve 2005 yılında da dağıtık sistem ile beraber üçüncü ensil kontrol sistemleri geliştirilmeye başlanmıştır.

Versiyon (Sürüm) Kontrol Sitemleri üç kategoriye ayrılır;

* Depo (Repository)
* Çakışma (Conflict)
* Dosya veya Dosya Kümesi İşlemleri

Github hesabı açıldı: <https://github.com/berfincinar>

* + 1. **Github Repository (Depo) Nedir?**



Repository (Depo), github da yaptığımız projelerin dosyalarını (img, txt vb.) depolandığı yerdir.

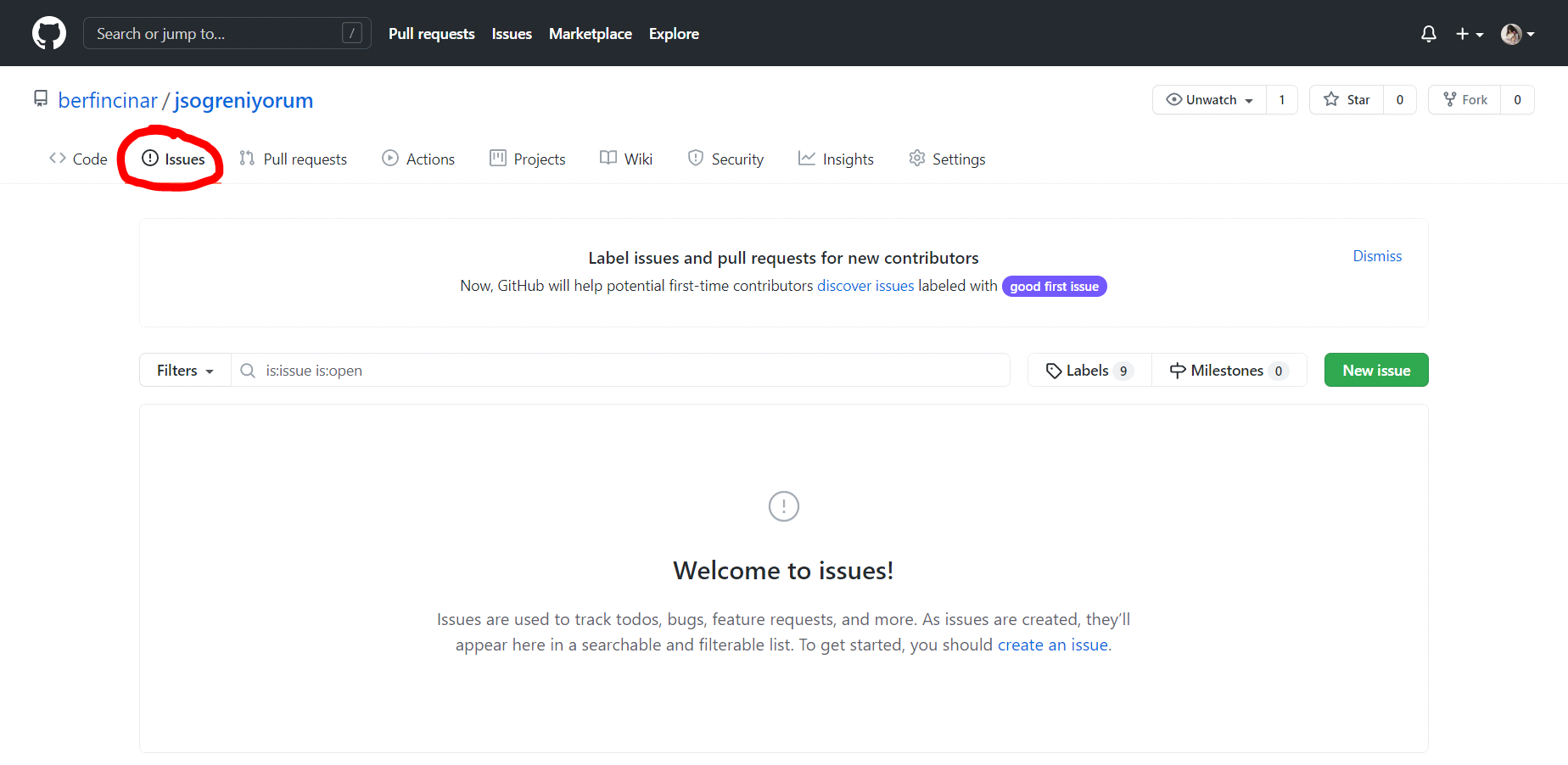
Github depomuz private ve public olarak ikiye ayrılır. Private, herkes tarafından ulaşılamayan demektir. Kullanıcı izin vermediği sürece hiç kimse kodlarına ulaşamaz. Public ise herkes tarafından ulaşılabilen depodur.

* + 1. **Github Contributor Nedir?**

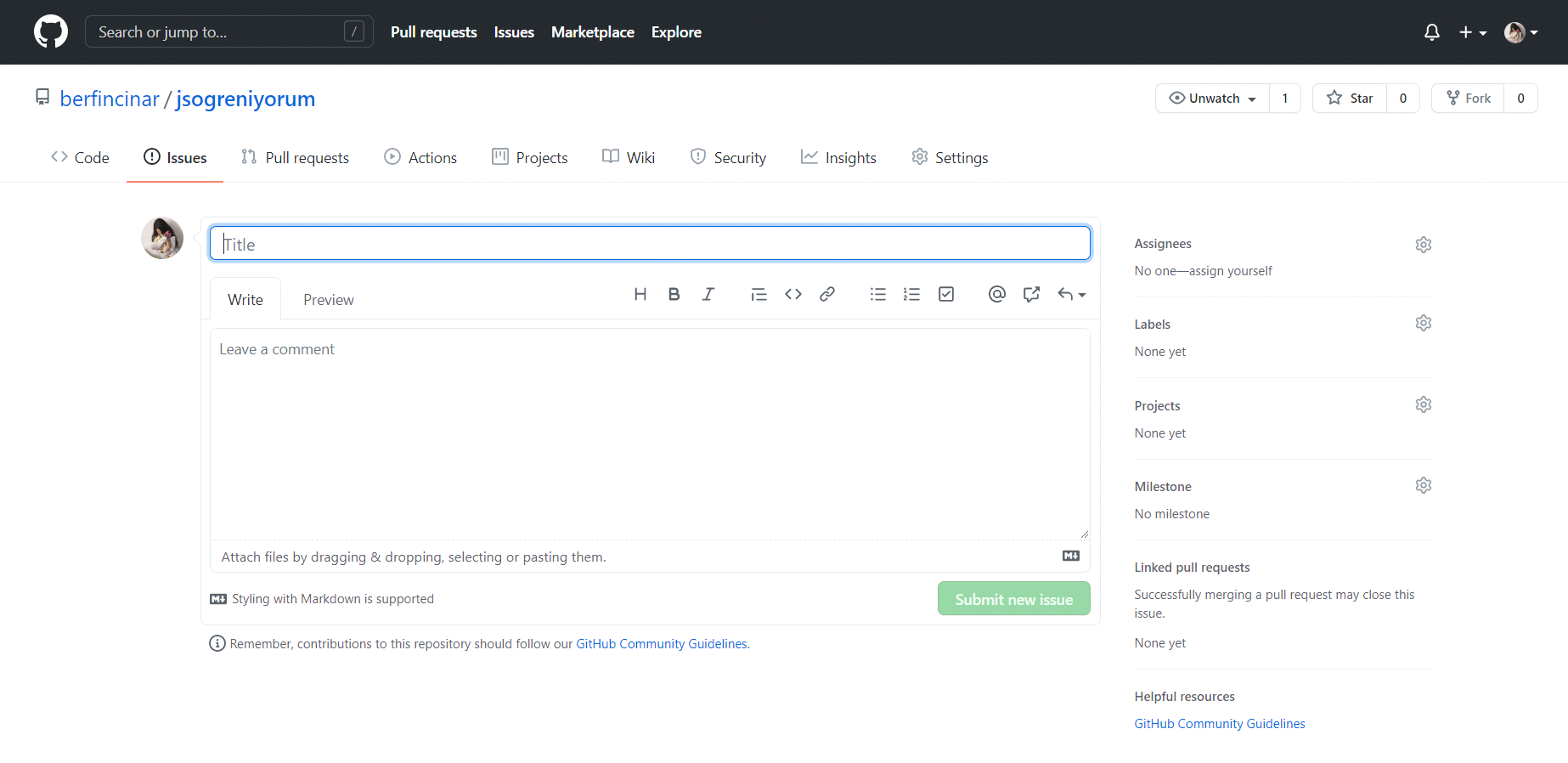
Github’ın bileşenleridir.

**3.1.6 Github Issue Nedir?**

Github görev yönetimi (issue) dir. Issue yapılacak işleri, sorunları kaydedip listeleyebileceğimiz bir yönetim ekranıdır.



Issue’ler yaptığımız projelerin görevlerini, geliştirmelerini ve hatalarını izlemek için kullanılan bir yoldur.



Issue girmek için “new issue” butonuna tıklarız ve yukarıdaki ekran önümüze çıkar. Talimatları inceleyerek yeni görevlerimiz, talimatlarımızı gireriz ve issue’larımız hazır olur.

**3.1.7 Github Commit Nedir?**

Commit’in Türkçe karşılığı “işlemek”tir. Commit’in github dilinde anlamı ise; Git’e eklediğimiz dosyalarımızı kalıcı olarak git’in veri tabanına işlenmesidir.

Komut sistemini kullanarak git veri tabanına dosyalarımızı eklemek için aşağıdaki komutu kullanırız.

git commit -m “commit mesajı”

Bu komut ile yapılan değişiklikler local depoya kaydedilir.

**3.1.8 Github Pull Request Nedir?**

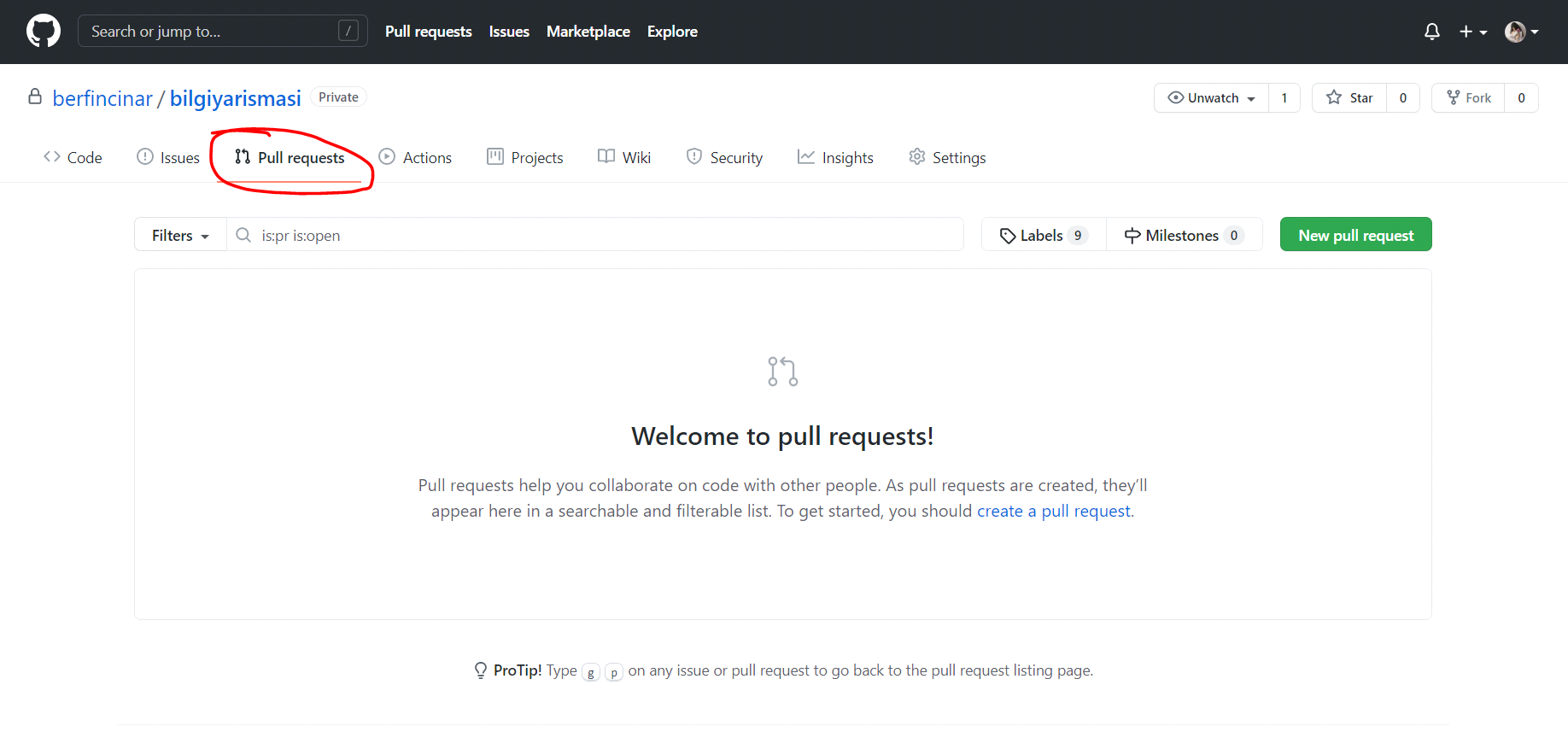
Github da fork ve pull request olarak iki özellik bulunmaktadır.

Fork, üzerinde çalıştığımız projenin kopyasını almak ve o kopya üzerinden çalışmak istersek kullanabileceğimiz bir özelliktir.

Pull Request ise; Fork ettiğimiz projenin üzerinde çalışıp, değişiklik yaptığımız halini proje sahibine “pull request” olarak gönderebiliriz.

Pull Request olarak göndermek demekse, proje sahibine “senin projen üzerinde değişiklikler yaptım, yapılan değişiklikleri kontrol edip onayla” demektir. Bu sayede onun projesine katkıda bulunmuş oluruz.

Pull Request dediğimiz olayına bir projede, proje sahibine yardım etmek diyebiliriz.

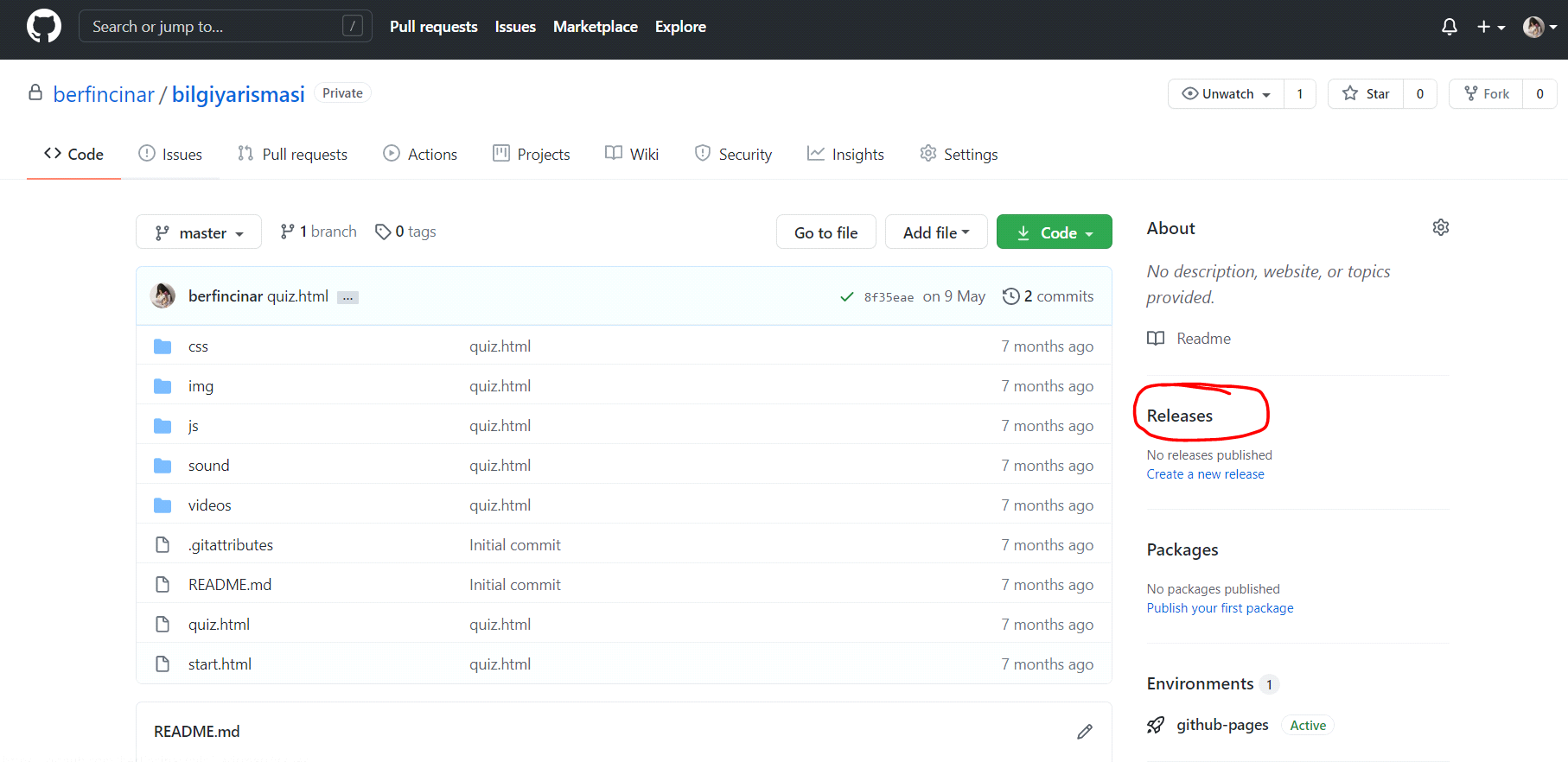


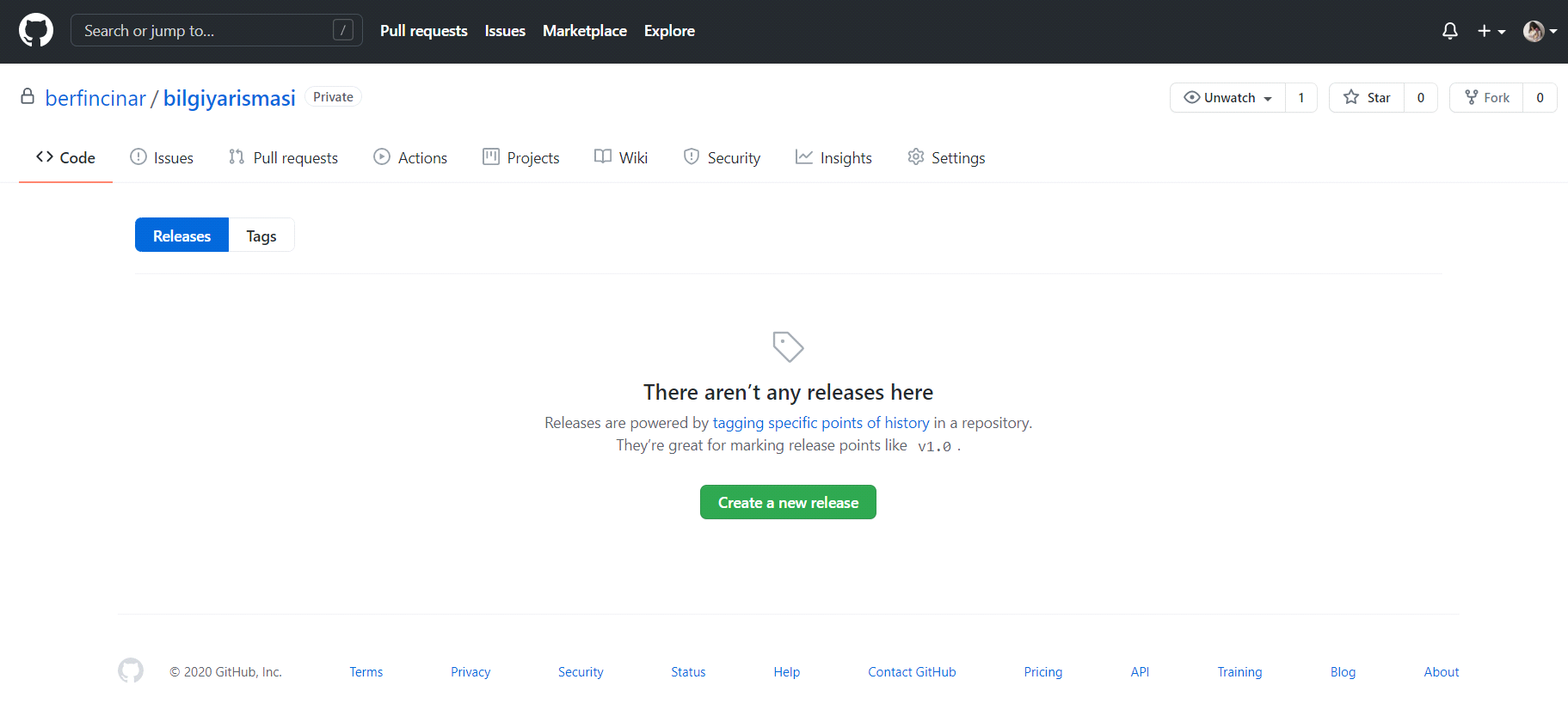
**3.1.9 Github Release Nedir?**

Başkalarının kullanması için sürüm notları ve ikili dosyalara bağlantılar ile birlikte yazılımı paketlemek için bir sürüm oluşturabilirsiniz.

Sürümler, paketleyip indirip kullanması için daha geniş bir kitleye sunabileceğiniz dağıtılabilir yazılım yinelemeleridir.

Yayınlar, kod deponuzun geçmişinde belirli bir noktayı işaretleyen Git etiketlerine dayanır. Bir etiket tarihi, farklı zamanlarda oluşturulabildiğinden, bir yayın tarihinden farklı olabilir.





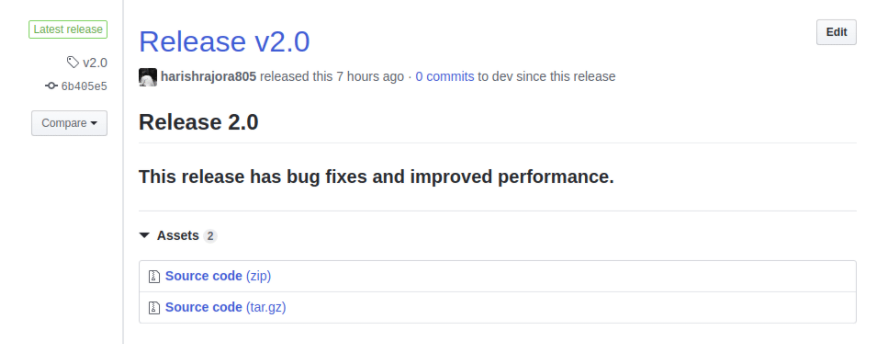
Release ile yapmış olduğunuz projeleri yayımlayabilirsiniz.

Aynı zamanda release yapacaksak branch ile çalışmak zorundasınız. Release , branch olmadan çalışmaz diyebiliriz.

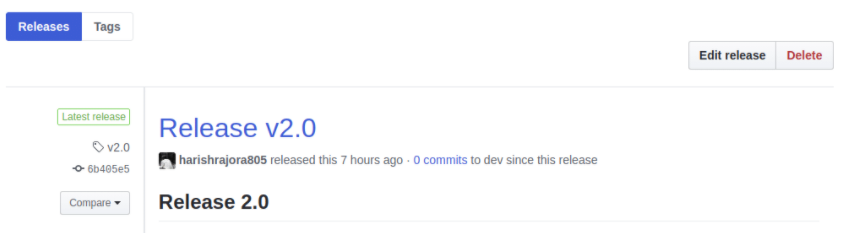
**GitHub'da Release Nasıl Silinir?**

GitHub'da bir release’i silmek oldukça basit bir iştir. Bir release’i silmek için aşağıdaki adımları izlemeniz yeterlidir. Bu örnekte v2.0 release’si sileceğiz :

1. GitHub hesabınızdan Releases sekmesine gidin.
2. Releases listesi ekranda görünecektir. Listeden Release v2.0 adlı sürümü seçin.

****

3-Release i seçtiğinizde, sürümün ayrıntıları açılacaktır. Sağda Sil butonu görünecektir.



4-GitHub sizden silme işlemini onaylamanızı isteyecektir.



5-Sürümü tamamen silmek için Bu sürümü Sil'e basın.

**GitHub Yayınlarında Sık Sorulan Sorular**

**Release oluşturmak gerekli mi?**

Releases, kullanıcılarınıza yazılımın en son sürümü hakkında bilgi vermenin mükemmel bir yoludur. Sürümler ayrıca büyük binary dosyaların depolanmasını önlemeye de yardımcı olur. Gerekli değildir, ancak iyi bir uygulamadır.

**GitHub Releases Git Taglarına mi dayanıyor?**

Evet, GitHub sürümlerinin temeli Git taglarıdır. Başka bir deyişle, bir yayın yayınlamak için tagların tanımlanması gerekir.

## **3.2. Release Notlarının Araştırılması**

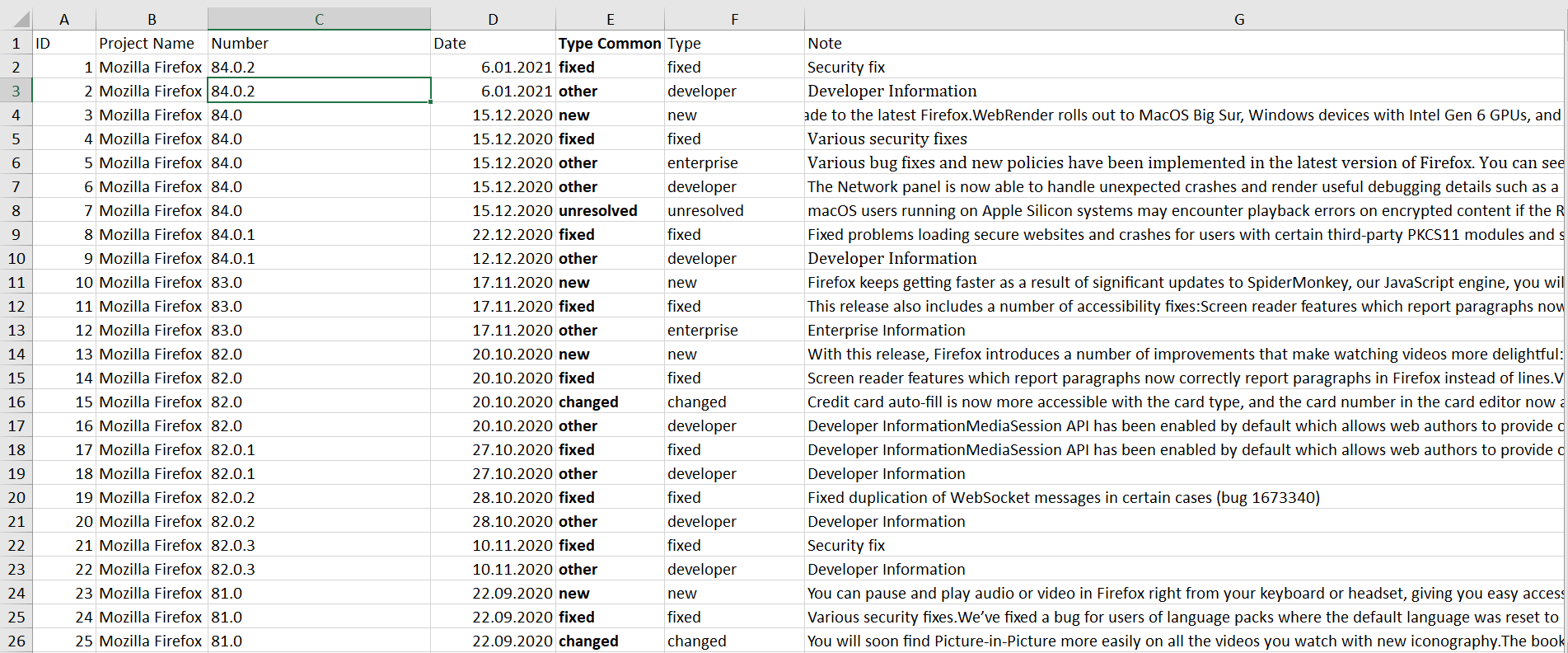
### **3.2.1. Mozilla Firefox Release Notları**

Mozilla Firefox Release notları <https://www.mozilla.org/en-US/firefox/releases/> internet sitesindedir. Burada 85.0’dan başlayıp 0.1’e kadar sürümler ilerlemektedir. Buradaki sürümler Python programlama diliyle birlikte Selenium araçlarını kullanarak otomatik bir şekilde worde kaydedildi. Kaydedilen dosya aşağıdadır **(Dosya üstüne çift tıklayınız )**;



Bu süreci gerçekleştiren programın kodları aşağıdaki gibidir;

### 



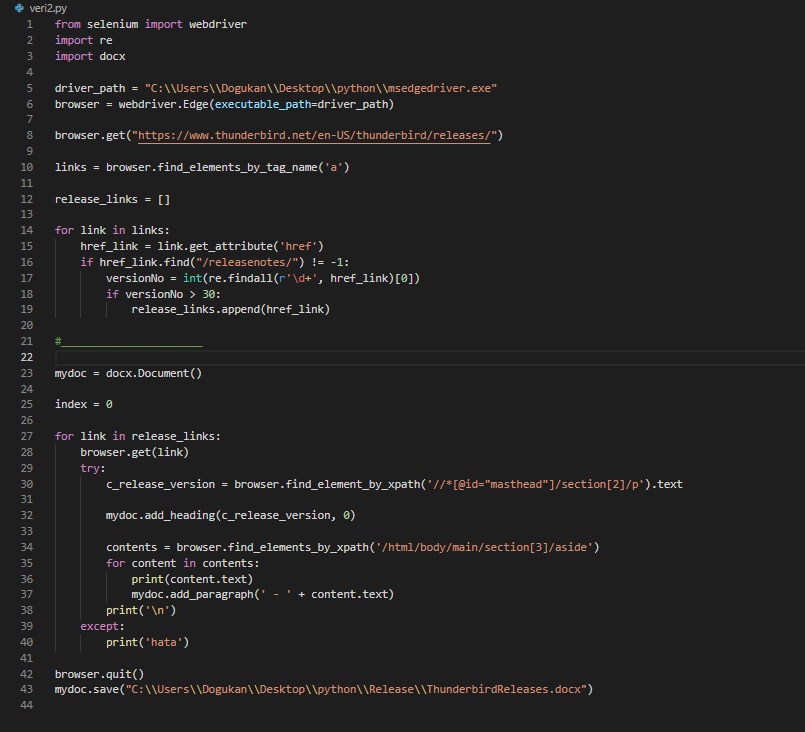
Buradaki release notlarından hareket edilerek diğer release notlar ile ortaklaşa bir veri seti oluşturulmaya çalışıldı ve bu veriler excel dosyasına döküldü. Ortak alanlar şu şekilde belirlendi ; fixed, new, changed, unresolved. Eğer release notundaki başlık bu ortak alanlardan biri değil ise other başlığı altında toplandı. Excel dosyasındaki alanlar; ID (1’den başlayarak sırasıyla arttı) , Project Name (Release Notundaki Proje İsmi), Number ( Release notundaki sürüm), Date (Yayınlandığı Tarih), Type Common (Diğer Projelerle Ortak Alan), Type(Değişiklik Başlığı) ve Note (Değişiklik Notu) olmak üzere 7 sütundan meydana geldi. 200 adet release note bu standartlara bağlı kalınarak excel dosyasına geçirildi.

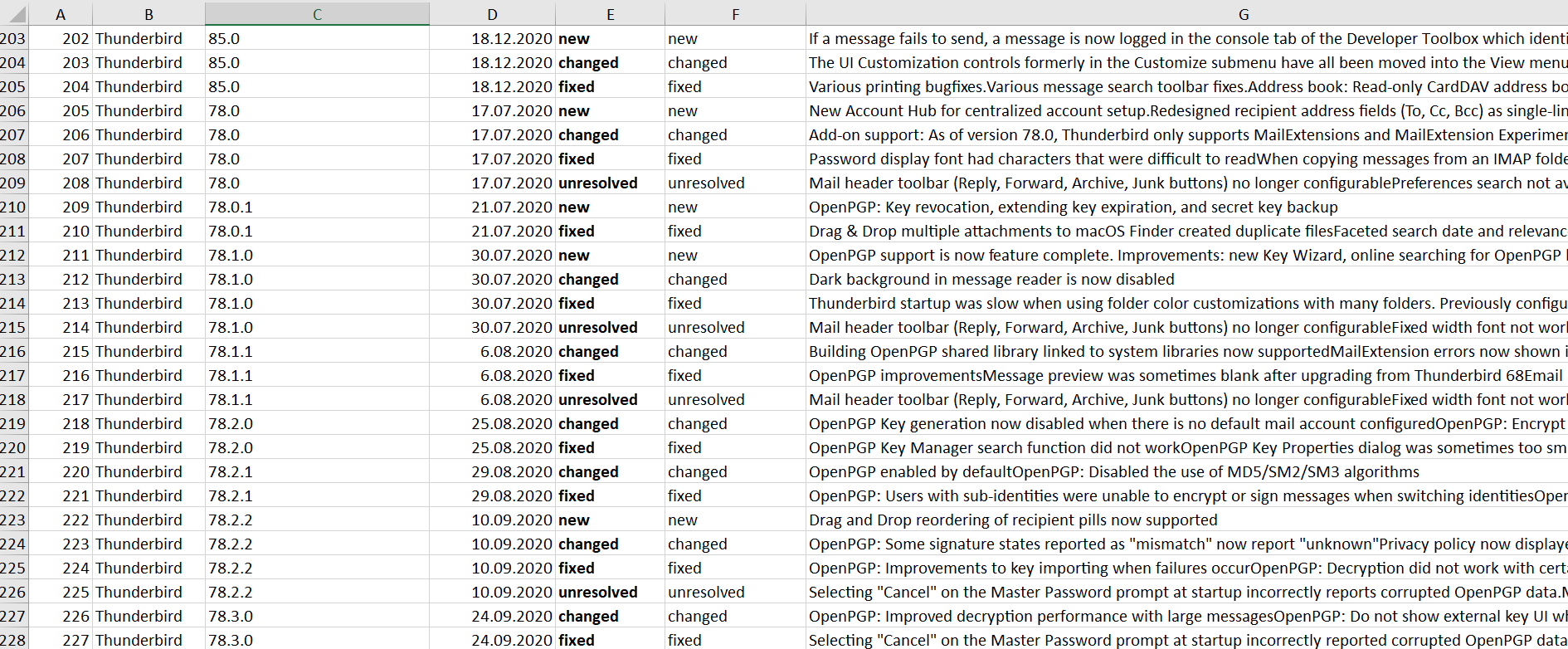
### **3.2.2. Mozilla Thunderbird Release Notları**

Mozilla Thunderbird Release notları <https://www.thunderbird.net/en-US/thunderbird/releases/> internet sitesindedir. Burada 78.0’dan başlayıp 0.1’e kadar sürümler ilerlemektedir. Buradaki sürümler Python programa diliyle birlikte Selenium araçlarını kullanarak otomatik bir şekilde worde kaydedildi. Kaydedilen dosya aşağıdadır **( Dosya üstüne çift tıklayınız )** ;



Bu süreci gerçekleştiren programın kodları aşağıdaki gibidir;





Buradaki release notlarından hareket edilerek diğer release notlar ile ortaklaşa

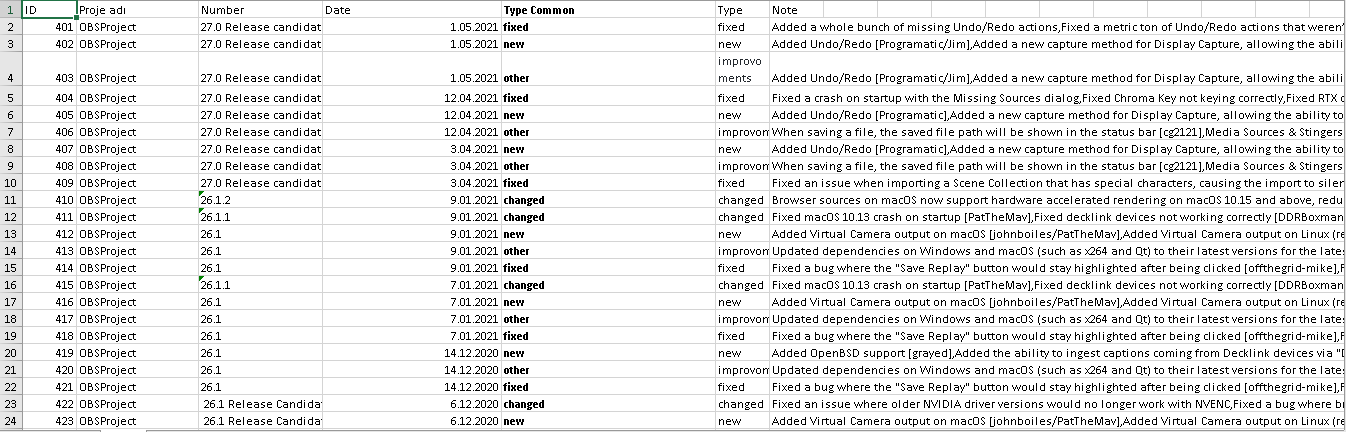
bir veri seti oluşturulmaya çalışıldı ve bu veriler excel dosyasına döküldü. Ortak alanlar şu şekilde belirlendi ; fixed, new, changed, unresolved. Eğer release notundaki başlık bu ortak alanlardan biri değil ise other başlığı altında toplandı. Excel dosyasındaki alanlar; ID (1’den başlayarak sırasıyla arttı) , Project Name (Release Notu ndaki Proje İsmi), Number ( Release notundaki sürüm), Date (Yayınlandığı Tarih), Type Common (Diğer Projelerle Ortak Alan), Type(Değişiklik Başlığı) ve Note (Değişiklik Notu) olmak üzere 7 sütundan meydana geldi. 200 adet release note bu standartlara bağlı kalınarak excel dosyasına geçirildi.

### **3.2.3. OBS Studio Release Notları**

OBS Studio release notları (obsproject/obs-studio) **https://github.com/obsproject/obs-studiointernet** sitesindedir. Burada 27.0 Release candidate 3 ’dan başlayıp 0.3.1’e kadar sürümler ilerlemektedir. **(Dosya üstüne çift tıklayınız )**;



Buradaki release notlarından hareket edilerek diğer release notlar ile ortaklaşa bir veri seti oluşturulmaya çalışıldı ve bu veriler excel dosyasına döküldü. Ortak alanlar şu şekilde belirlendi ; fixed, new, changed, unresolved. Eğer release notundaki başlık bu ortak alanlardan biri değil ise other başlığı altında toplandı. Excel dosyasındaki alanlar; ID (1’den başlayarak sırasıyla arttı) , Project Name (Release Notundaki Proje İsmi), Number ( Release notundaki sürüm), Date (Yayınlandığı Tarih), Type Common (Diğer Projelerle Ortak Alan), Type(Değişiklik Başlığı) ve Note (Değişiklik Notu) olmak üzere 7 sütundan meydana geldi. 317 adet release note bu standartlara bağlı kalınarak excel dosyasına geçirildi.



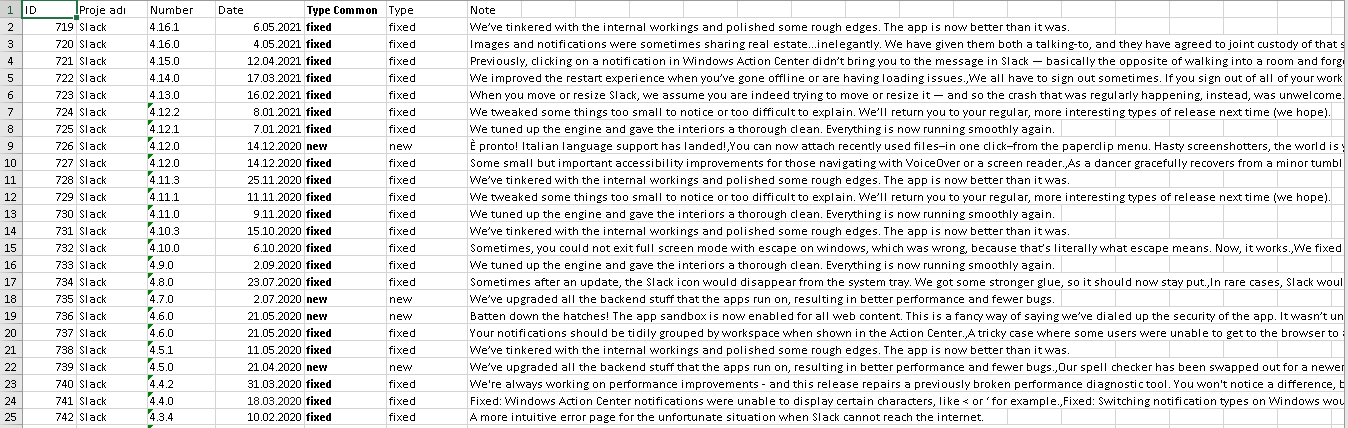
### **3.2.4. Slack Release Notları**

Slack release notları **https://slack.com/intl/en-tr/release-notes/windows** sitesindedir. Burada 4.16.1 ’dan başlayıp 2.6.5’e kadar sürümler ilerlemektedir. **(Dosya üstüne çift tıklayınız )**;



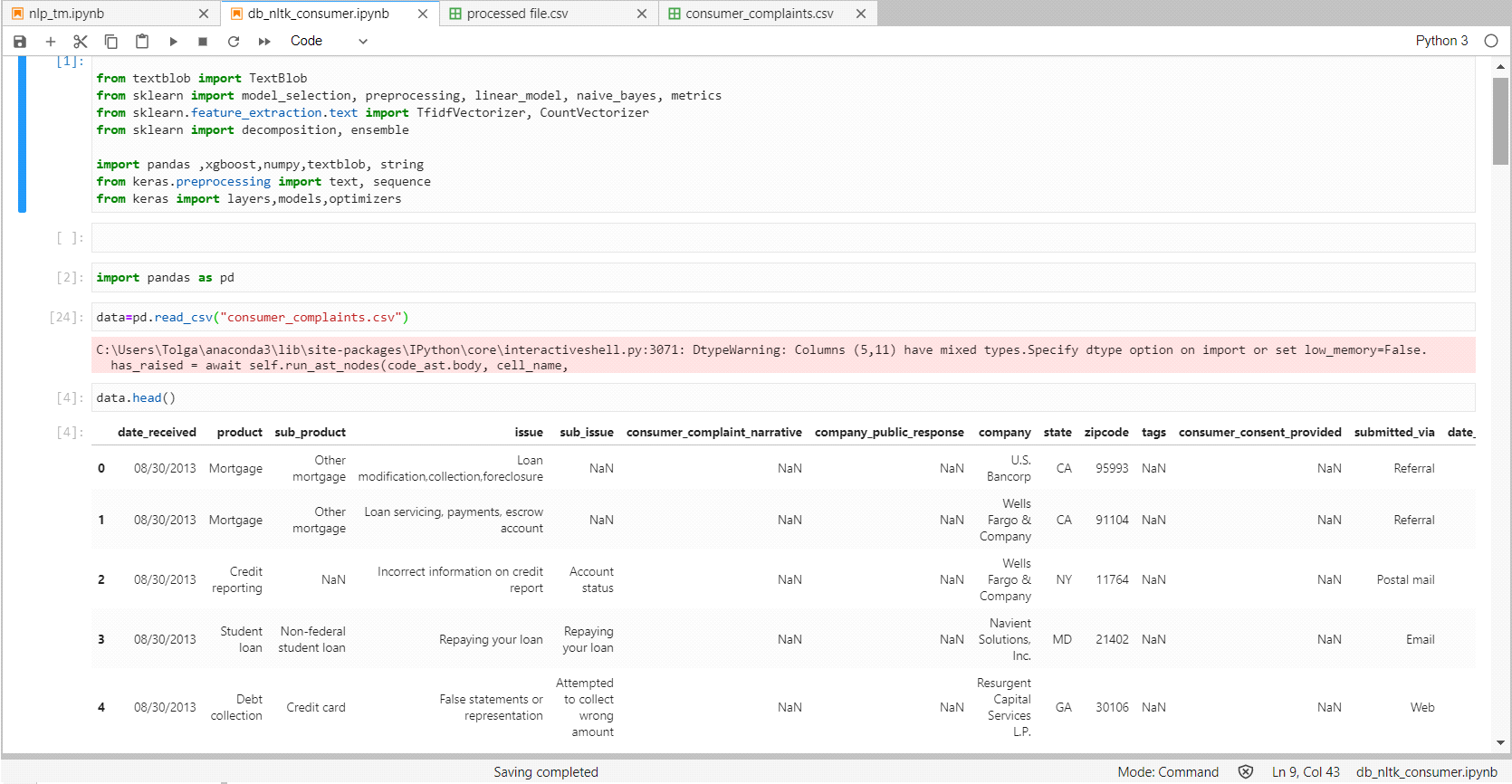
Buradaki release notlarından hareket edilerek diğer release notlar ile ortaklaşa

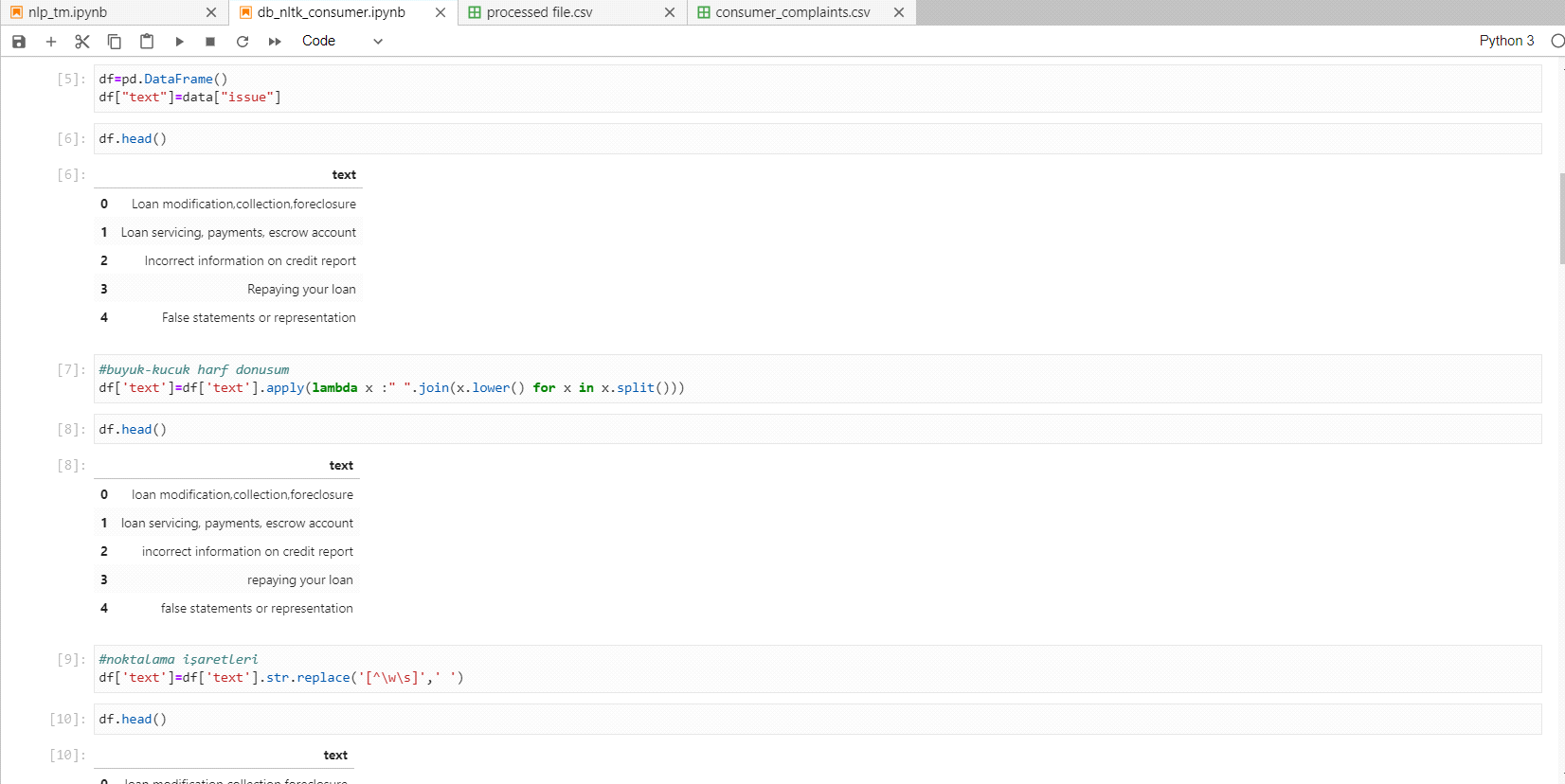
bir veri seti oluşturulmaya çalışıldı ve bu veriler excel dosyasına döküldü. Ortak alanlar şu şekilde belirlendi ; fixed, new, changed, unresolved. Eğer release notundaki başlık bu ortak alanlardan biri değil ise other başlığı altında toplandı. Excel dosyasındaki alanlar; ID (1’den başlayarak sırasıyla arttı) , Project Name (Release Notu ndaki Proje İsmi), Number ( Release notundaki sürüm), Date (Yayınlandığı Tarih), Type Common (Diğer Projelerle Ortak Alan), Type(Değişiklik Başlığı) ve Note (Değişiklik Notu) olmak üzere 7 sütundan meydana geldi. 70 adet release note bu standartlara bağlı kalınarak excel dosyasına geçirildi.

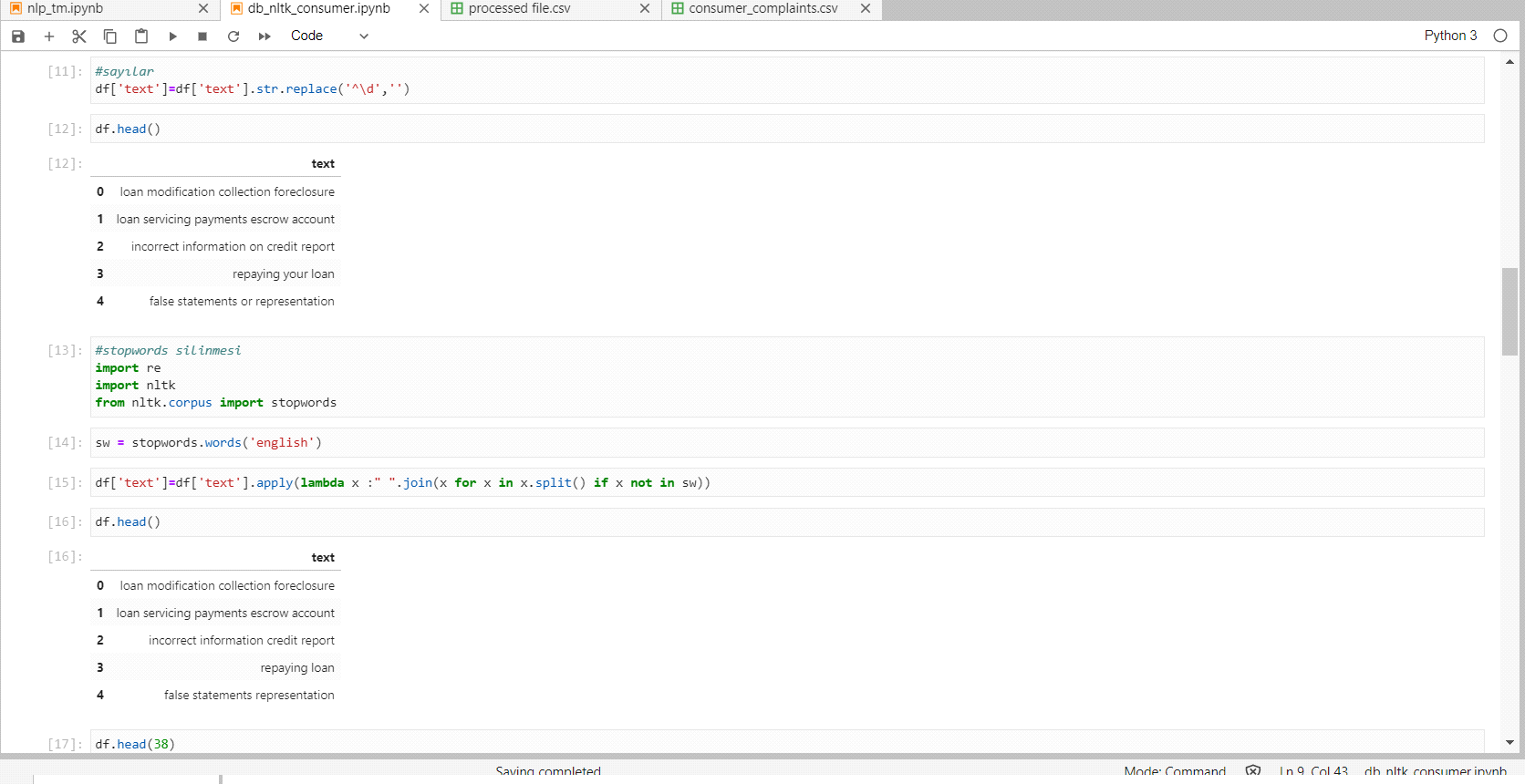


**3.3.Text Pre Processing İşlemleri**

**3.3.1. Text Pre Processing**





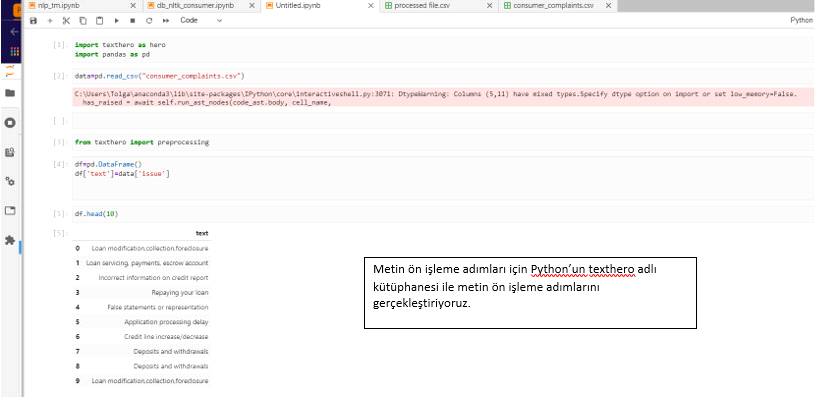


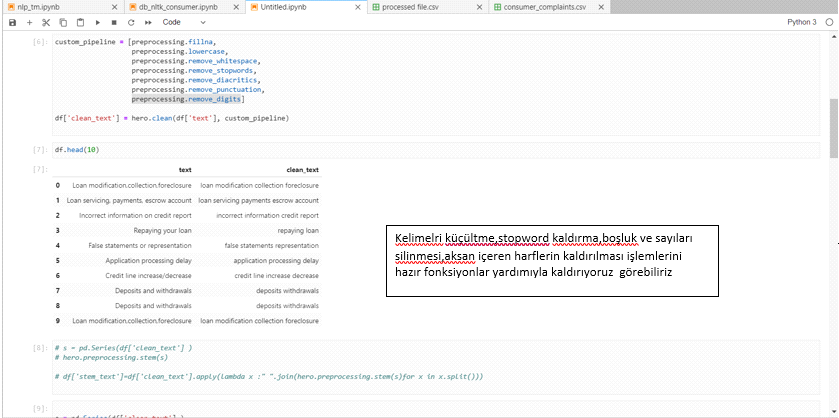
Data set üzerinde text pre processing işlemleri :sayıların ,noktalama işaretlerinin az gecen kelimelerin ve eklerin temizlenmesi ,kelime köklerinin bulunması .Yukarıdaki 3 fotoğrafta adım adım işlenmiştir son görselde ise elde edilen sonuç görünmektedir elimizdeki data set 55500 satırdan oluştuğu için detaylı gösterim için kodlara bakınız

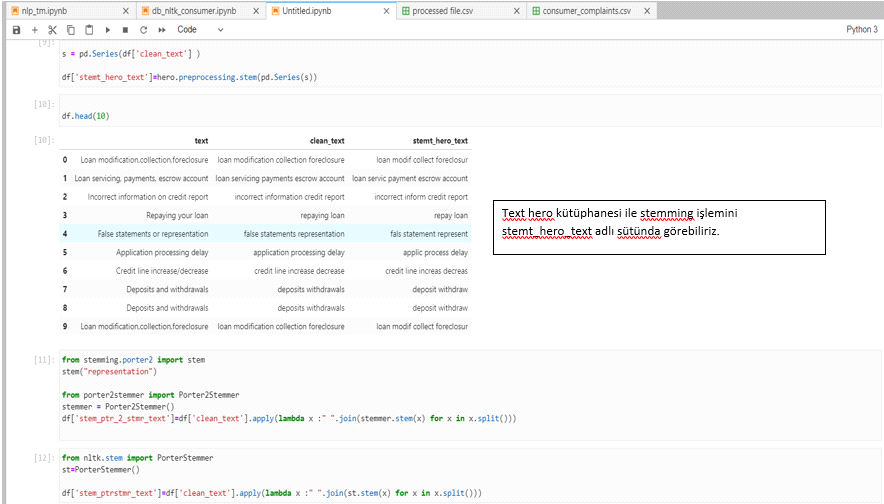


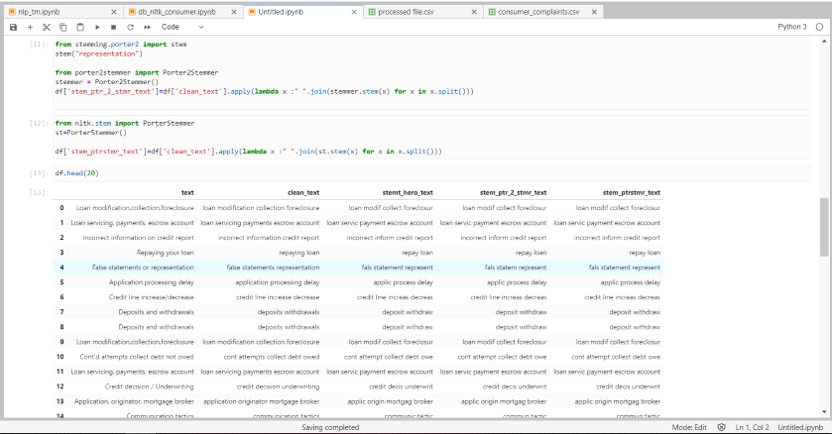
SONRASI/ÖNCESİ

**3.3.2. Text Hero İle Text Pre Processing**









Son olarak Porter2Stemmer ve PorterStemmer kullanarak text hero ile karşılaştırmamınız yukarıdaki görselde görebiliriz.

**3.3.3. Stemming**

Kelime köküne erişmek için genel olarak iki yöntem vardır: stemming ve lemmatization. Her iki işlemin amacı da aynıdır: her kelimeyi eklerinden arındırıp ortak bir temel ya da köke dönüştürmek.

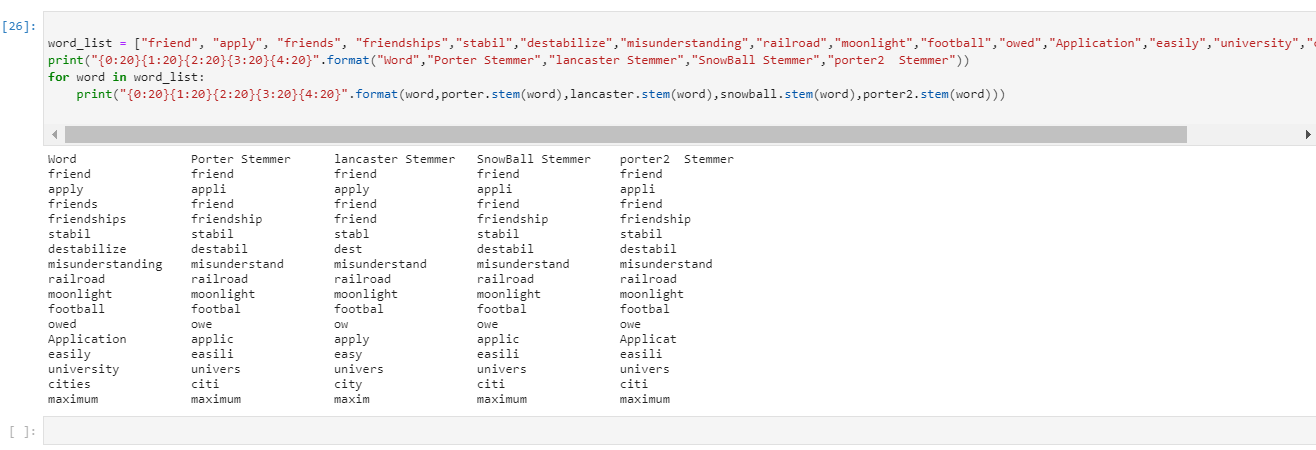
Hem stemming hem de lemmatization, genellikle veri ön işleme adımlarındandır ve aşina olmanız gereken şeylerdendir.

Stemming algoritmaları, ekli bir kelimede bulunabilen ortak ön eklerin ve son eklerin bir listesini dikkate alarak kelimenin başlangıcını veya sonunu kesmeye çalışır.

#### Stemming Algoritma Örnekleri

* *Porter stemmer:* Bu stemming algoritması en eski stemming algoritmalarından biridir. Asıl kaygısı, ortak bir biçimde çözülebilmeleri için kelimelerin ortak sonlarını çıkarmaktır. Çok karmaşık değildir ve artık geliştirimi sürmemektedir. Genellikle başlangıç için iyidir, ancak herhangi gerçek bir uygulama için kullanılması tavsiye edilmez. Ayrıca, diğerlerine kıyasla çok nazik bir stemming algoritmasıdır.
* *Snowball stemmer:* Bu algoritma aynı zamanda Porter2 stemming algoritması olarak da bilinir. Neredeyse herkes tarafından Porter stemmer'dan daha iyi olduğu kabul edilir. Aynı zamanda Porter stemmer'dan daha agresiftir. Snowball stemmer'ına eklenen şeylerin çoğu, Porter stemmer'ında fark edilen konulardan kaynaklanmıştır.
* *Lancaster stemmer:* Lancaster stemming algoritması kullanılabilecek bir başka algoritmadır. Bu, en agresif stemming algoritmalarından biridir. Ancak, NLTK'da kullanımında, bu algoritmaya özel kuralların eklenmesi çok kolaydır. Bu stemming algoritması hakkındaki şikayetlerden biri, bazen aşırı saldırgan olduğu ve kelimeleri gerçekten garip stem'lere dönüştürebilmesidir

**3.3.4. Stemming Kütüphaneleri Karşılaştırılması**



Kelime köklerini daha rahat kıyaslayabilmek için elimiz de olan dört kütüphane üzerinden bir karşılaştırma uygulaması gerçekleştirdik.

Text hero stemming için nltk.stem.SnowballStemmer ve nltk.stem.PorterStemmer kullandığı için karşılaştırmada yer vermedik.

Resimde de görüldüğü üzere temel yapıda dahi lancaster ve snowball stemmer daha avantajlı çalıştığını söylemek mümkün.

Bu iki stemmer arasından bir seçim yapmak için ikisini de inceleyim.

Örnek olarak lancaster stemmer default\_rule\_tuple = (

"re2>", *# -er > -)* kuralından sebeple ‘meter’ kelimesini kökünü ‘met’ olarak buluyor.

(tüm kurallar bkz: <https://www.nltk.org/_modules/nltk/stem/lancaster.html#LancasterStemmer.parseRules>)

İngilizce exceed kelimesi ‘aşmak’ anlamındadır ve kelimenin kökü yoktur yani kökü kendisidir, lancaster stemmer bu kelimeyi excess ‘aşırı’ anlamındaki kelimeye kök olarak gösterdi ama snowball specialwords altında özel tanımlı bir kelime olduğu için düzgün çalıştı.

Aslında kelimelerin düzgün bulunamamasının sebebi stemmer algoritmalarının kuralları genelleme ile yazıldığı için bazı kelimeler hata verebiliyor lancater da bunu düzeltmek kural oluşturma yapısı ile mümkün.

st\_custom = LancasterStemmer(rule\_tuple=("ssen4>", "s1t."))

st\_custom.stem("ness")

-nest

İki stemmer da Cried, paid, examined sırası ile cry ,pay, examined olması gerekirken cri,paid, examin olarak kök buldu

**Lancaster :**

"e1>", *# -e >-,*

i\*1.", *# -i > - if intact,*

1y>", *# -i >*

Kuralları ile cities kelimesinin kökünü doğru bulurken,

Snowball citi olarak buluyor:

**elif** suffix **in** ("ied", "ies"):

**if** len(word[: -len(suffix)]) > 1:

word = word[:-2]

r1 = r1[:-2]

r2 = r2[:-2]

yukarıda gördüğümüz gibi ied ve ies takısını kelimenin sonundan iki harf silerek yapıyor ve i’nin y’ye dönüşme işlemi yapılmıyor.

bkz: [https://www.nltk.org/\_modules/nltk/stem/lancaster.html#LancasterStemmer.parseRules](https://www.nltk.org/_modules/nltk/stem/lancaster.html)

<https://www.nltk.org/_modules/nltk/stem/snowball.html#ArabicStemmer>

**Kullanılan Kelimeler:**

"examined, exceeds, apply, friends, friendships, stabil, destabilize, misunderstanding, railroad, moonlight, football, owed, Application, easily, university, cities, maximum"

**3.3.5. Porter2 Stemmer İle Veri Setinin Temizlenmesi**

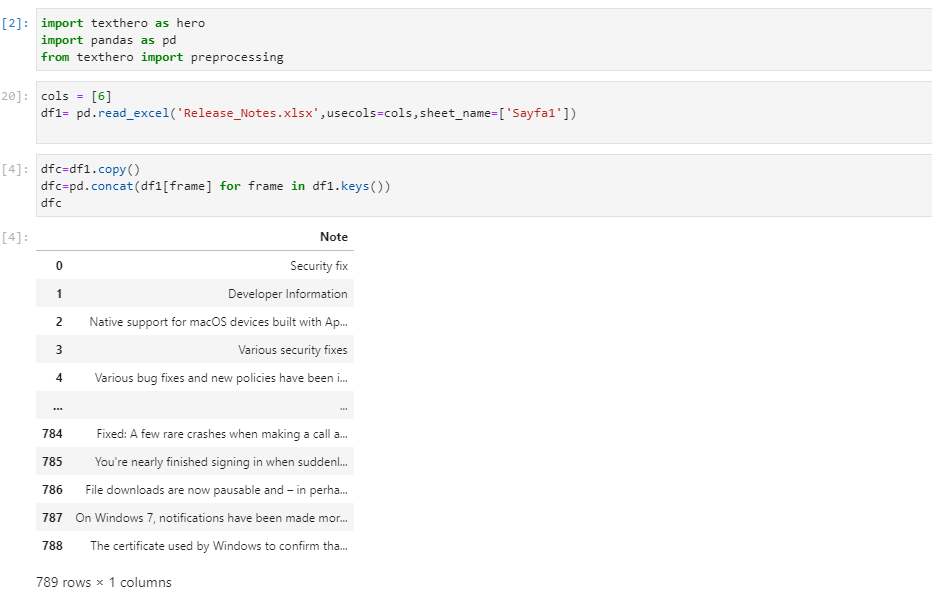
Öncelike şunlardan bahsetmek gerekiyor :

**. Kök ayırıcı neden uygun kelimeler üretmiyor?**  
  
Bir kök bulma algoritmasının, kökü çıkardıktan sonra gerçek bir kelime bırakmaması genellikle kaba bir hata olarak kabul edilir. Ama kökten çıkarmanın amacı, bir sözcüğün farklı biçimlerini bir araya getirmektir, bir sözcüğü 'paradigma' biçimiyle eşleştirmek değil.  
  
Ve bununla bağlantılı,

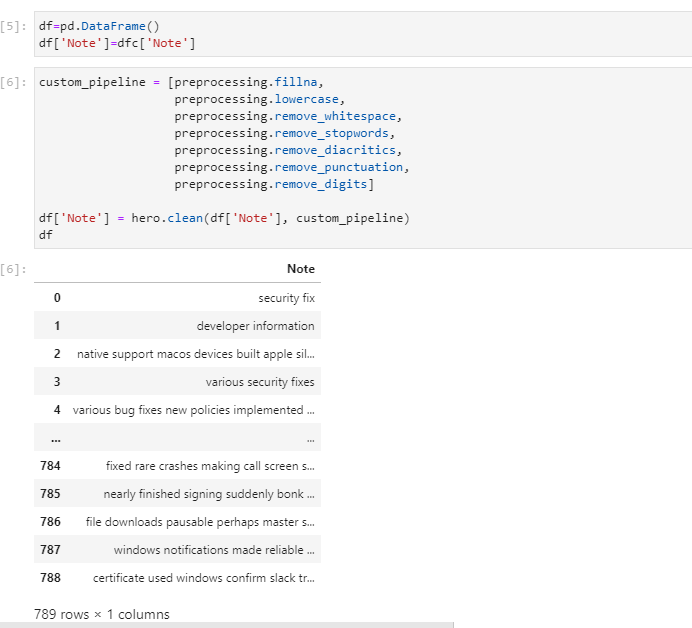
**Neden hatalar var?**  
  
Soru normalde şu şekilde gelir: *X* kelimesinin kökü neden *x1 olsun ki* , insan onun *x2* ile köklenmesi beklenirken? Kök alma algoritmasının mükemmelliğe ulaşamayacağını hatırlamak önemlidir. Dengede, IR performansını artıracaktır (veya iyileştirebilir), ancak bazı durumlarda bazen veya görünen hataları yapabilir. Tabii ki, bu, Stemmer'ın performansını artırmak için dahil edilebilecek ek bir kural önermekten farklı bir konudur.

## Bu ne tür bir saptırıcı?

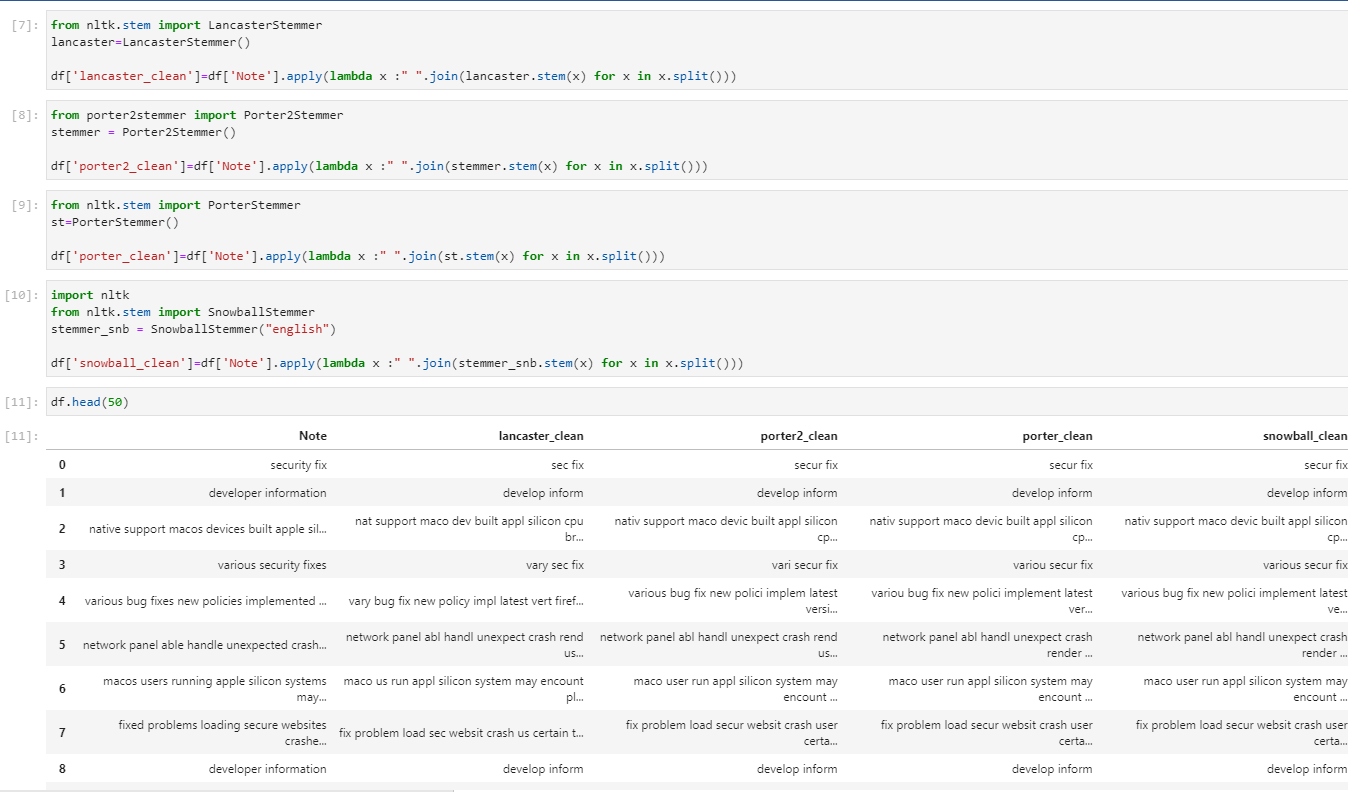
Porter2 bir sonek soyucu kök ayırıcıdır. Sözcüğün son kısmına deterministik bir değişiklik dizisi uygulayarak sözcükleri köklere dönüştürür. Diğer kök ayırıcılar farklı çalışır. Örneğin, bir tablodaki çekimli forma bakıp onu morfolojik bir kökle eşleyebilirler veya çeşitli formları bir merkez forma eşlemek için bir kümeleme yaklaşımı kullanabilirler. Farklı yaklaşımların farklı avantajları ve dezavantajları vardır.



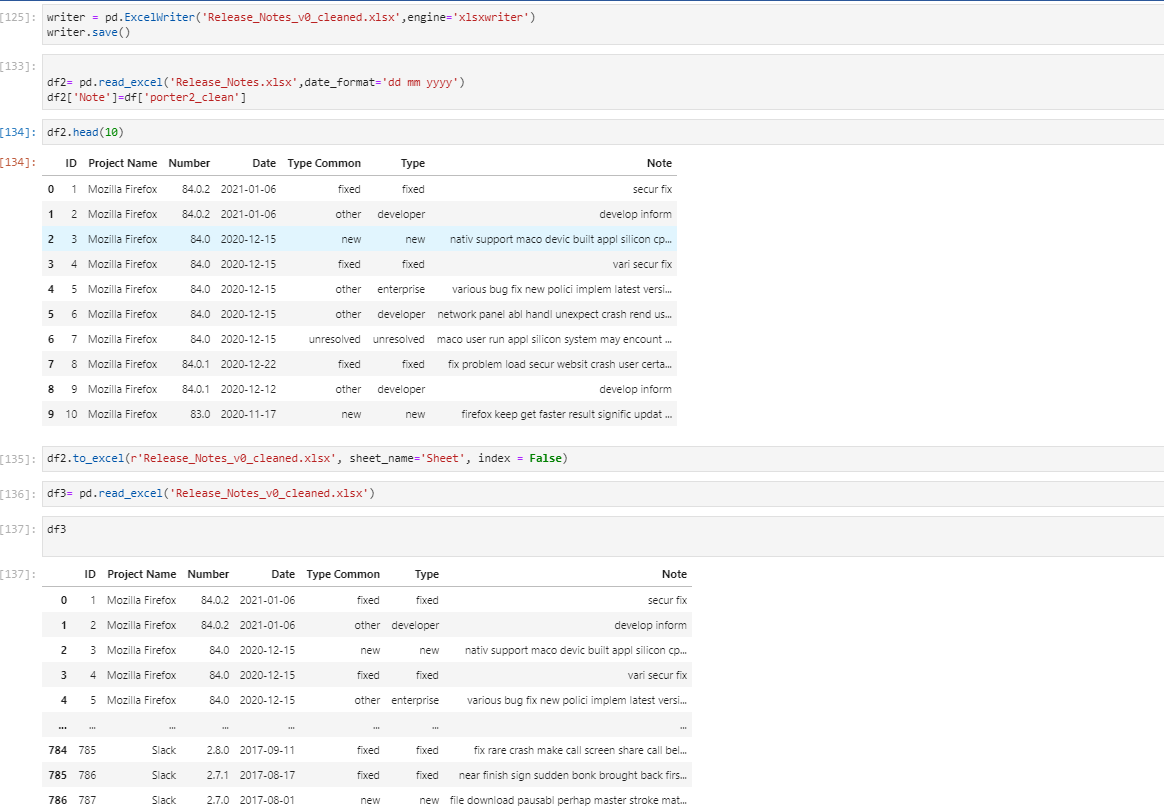
Text hero ile veri setimizi temizleme işlemlerini gerçekleştirmeden bir data frame oluşturup buradan devam etmemizin uygun olacağını düşündük.



Text hero text pre processing yaparken apply fonkisyon ile metini kelime kelime dolanmak ve her tür temizleme için ayrı uğraşmaktan cok daha kolay bir kullanım sağlıyor bknz(yukardaki resim)



Zaten başından beri bizi en çok zorlayan ,düşündüren durum hangi stemmer algoritması? Sorusuydu burada yine karşılaştırma ile son karar vermemiz gerekti yukarıda da bahsettiğimiz gibi porter2 stemmer’ın sertliği ve ortak kelime coğunlu sağlaması bizim için çok avantajlı bir duurm olduğuna karar verdik son çıktımızı porter2stemmer ile aldık



Son olarak veri setimizi yapay zeka algoritmamıza hazır hale getirip çıktısı aldığımız işlemler yukarıda.

**3.4 Makine Öğrenmesi**

**3.4.1Makine Öğrenmesi Temel Kavramları**

Supervised Learning(Gözetimli Öğrenme) : Vereceğimiz girdilere karşılık hangi çıktıyı elde edeceğimizi algoritmamıza/makineye söylüyorsak bu gözetimli öğrenmedir.

**3.4.1.1Feature(Özellik) ve Label(Etiket) Nedir?**

Label veri setimizde tahmin edilmesini istediğimiz kolon değeridir. Feature ise makine öğrenmesi modelinin tahmin yapabilmesi için verdiğimiz girdi değeridir.

Elimizde bir evin değerini bulan modelimiz olsun. Modelimiz evin değerini bulmak için

* Toplam oda sayısı (feature)
* Metrekare(feature)
* Konum(feature)
* Bina Yaşı(feature)

gibi bilgileri kullanacak. Bunları modelimize veriyoruz, aynı zamanda eğitim veri setimize verdiğimiz bu değerlere karşılık her ev için “evin değerini”(label) veriyoruz. Bunları bir arada kullanıdığımızda example(örnek) elde ediyoruz.

|  |  |  |  |
| --- | --- | --- | --- |
| **BinaYasi(Feature)** | **ToplamOdaSayisi(Feature)** | **MetreKare(Feature)** | **EvFiyatı(Label)** |
| 15 | 3 | 120 | 200000 |
| 19 | 4 | 150 | 250000 |
| 24 | 5 | 175 | 275000 |

**3.4.1.2. Model Nedir?**

Model Feature ve Label arasındaki ilişkiyi tanımlar.

* **Training(Eğitim)** modeli oluşturmak veya modeli öğrenmek anlamına gelir. Yani modele Label örneklerini gösteririz ve modelin feature ile label arasındaki ilişkiyi zamanla öğrenmesini sağlarız.
* **Inference(Çıkarım)** ise eğitilmiş modeli unlabeled(etiketlenmemiş) örneklere uygulama anlamına gelir.

**3.4.1.3. Regresyon ve Sınıflandırma**

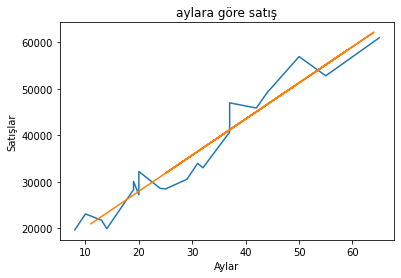
Regresyonda modelimiz sürekli veriler üzerinde tahminde bulunur. Sayısal çıktılar elde ederiz.

Örneğin;

* Sivas’ta bir evin değeri nedir?
* Bir kullanıcının reklama tıklama olasılığı nedir?

**3.4.1.4. Linear Reggresion(Lineer Regresyon)**

Lineer Regresyonda modelimiz eğitim setindeki verilerimiz üzerinde doğrusal bir ilişki yakalamaya çalışmaktadır.



**Resim-5**

Yukarıdaki doğrusal çizgi bizim modelimiz. Verileri gözlemledi ve her birine olabildiğince yakın durmaya çalıştı ve bir doğrusallık yakaladı.

Matematiksel olarak ifade edersek:

y= mx + b

* y : elde etmeye çalıştığımız tahmin değerimiz
* m : eğim
* b : y kesme noktası

ML’de ise bu eşitlik şu şekilde gösterilir:

yI = b + w1x1

* yI  = tahmin edilen label(istenen çıktı)
* b = bias’tır(y’yi kesme noktası), bazen w0 olarakta kullanılır
* w1 = feature(özellik) ağırlığıdır. Ağırlık eğim ile kavram olarak aynıdır.
* x1 = feature(özellik)

Eğer birden fazla özelliğimiz(feature) varsa birden fazla ağırlığımız olacaktır.

yI = b + w1x1 + w2x2 + w3x3

**3.4.1.5. Training ve Loss (Eğitim ve Kayıp)**

Denetimli öğrenmede(Supervised Learning) bir ML algoritması birçok örneği inceleyerek ve kaybı en aza indiren bir model bulmaya çalışarak bir model oluşturur.Bu sürece empirical risk minimization denir.

Loss(kayıp) ise, modelin tahmininin ne kadar kötü olduğunu gösteren bir sayıdır. Modelin tahmini mükkemelse Loss sıfırdır tam tersi durumda ise Loss değeri daha büyüktür.

**3.4.1.6. Train ve Test Kavramları**

Modelimizin loss değeri ne kadar az ise o kadar iyi diyebiliriz ancak modelimiz veri setimizi ezberlemiş olabilir. Bu duruma **overfitting** denir. Yani modelimiz eğitim aşamasında çok güzel tahminlerde bulunuyor fakat modelimize daha önce görmediği, eğitim setinde bulunmayan bir veri verdiğimizde yanlış tahminlerde bulunuyorsa overfitting durumuna gelmiştir denilebilir.

Bunun önüne geçmek için verimizi iki parçaya bölüyoruz.

* **Train data :** Modelimiz kendini bu verileri kullanarak eğitecek.
* **Test data :** Eğitilmiş verileri kontrol etmek için test seti üzerinde de denemeler yapacak.

Train Data

Test Data

Training Set Test Set

**3.4.1.7.Logistic Regression(Lojistik Regresyon)**

Belirli bir sınıf veya olayın olasılığını modellemek için kullanılır. Kanser tedavisinde kullanılan bir modelin kanser ya da kanser değil gibi bir çıktı vermesi çok tehlikeli olacağından lojistik regresyon kullanabiliriz. Bu model bize 0-1 arasında değerler döndürecek.

Örnek vermek gerekirse; Kanser teşhisinde 0.5 değeri orta bir değer olsun. Bizim çıktımız ise 0.48 olsun. Eğer lineer regresyon kullansaydık bize vereceği cevap belkide kanser değil olacaktı ama lojistik regresyon bize %48 ihtimalle kanser olma ihtimali var diyecektir.

**Classification(Sınıflandırma)**

* Sınıflandırma ayrık veriler üzerinde çalışır.
* Sınıflandırma sıralanmamış, dağınık veriler üzerinde işlem yapar.
* Model değerlendirme yöntemlerinde sınıflandırma algoritmalarında **‘confussion matrix, accuaracy, recall, F1 score’** gibi değerlendirme yöntemleri kullanılır
* Sınıflandırmada kategorik verilerle(spam, spam değil, kanser, kanser değil) çalışılır.

**Threshold değeri nedir?**

Eşik değer dediğimiz threshold, algoritmamızın tahminlerinde kullanılır. Örneğin spam yada spam değil diye tahminlerde bulunabilen bir algoritmamız var ve bu algoritma 0-1 arasında değerler tahmin ediyor. 1 değeri = %100 spam , 0 değeri = %0 spam değil gibi. Eğer 0.48 ise ne yapacağız. Burada algoritmamıza “Eşik değeri olarak 0.5 al ve bundan büyük veya eşit olanaları “spam”, küçük olanları “spam değil” olarak bize döndür diyebiliriz.

**3.4.2. Makine Öğrenimi Algoritmaları Ve Karşılaştırmaları**

**3.4.2.1. Lineer Regresyon:**

Özellikleri alır ve sürekli bir çıktı öngörür. Örneğin: hisse senedi fiyatı, maaş vb. Adından da anlaşılacağı gibi doğrusal regresyon, her soruna doğrusal bir eğri çözümü bulur.

Avantajları:

* Kolay ve basit uygulama
* Hızlı eğitim

Dezavantajları:

* Yalnızca çözüm doğrusal ise uygulanabilir. Pek çok gerçek hayat senaryosunda, durum böyle olmayabilir.
* Algoritma, girdi artıklarının (hata) normal dağıldığını varsayar, ancak her zaman karşılanmayabilir.
* Algoritma, girdi özelliklerinin karşılıklı olarak bağımsız olduğunu varsayar (eş doğrusallık yoktur).

LR İçin Varsayımlar

* Bağımsız ve bağımlı değişkenler arasındaki doğrusal ilişki.
* Eğitim verilerinin homoskedastik olması, yani hataların varyansının bir şekilde sabit olması gerektiği anlamına gelir.
* Bağımsız değişkenler eş doğrusal olmamalıdır.

LR vs Karar Ağaçları

* Daha az veri setine sahip (düşük gürültülü) çok sayıda özellik olduğunda, doğrusal regresyonlar Karar ağaçlarından / rastgele ormanlardan daha iyi performans gösterebilir. Genel durumlarda, Karar ağaçları daha iyi bir ortalama doğruluğa sahip olacaktır.
* Kategorik bağımsız değişkenler için karar ağaçları doğrusal regresyondan daha iyidir.
* Karar ağaçları, eşdoğrusallığı LR'den daha iyi yönetir.

LR vs SVM

* SVM, çekirdek numarası kullanarak hem doğrusal hem de doğrusal olmayan çözümleri destekler.
* SVM, aykırı değerleri LR'den daha iyi yönetir.
* Her ikisi de eğitim verisi daha az olduğunda ve çok sayıda özellik olduğunda iyi performans gösterir.

LR vs KNN

* KNN parametrik olmayan bir modeldir, oysa LR parametrik bir modeldir.
* KNN, tüm eğitim verilerini takip etmesi ve komşu düğümleri bulması gerektiğinden gerçek zamanlı olarak yavaştır, oysa LR, ayarlanmış θ katsayılarından çıktıyı kolayca çıkarabilir.

LR vs Sinir Ağları

* Sinir ağları, LR modeline kıyasla daha büyük eğitim verilerine ihtiyaç duyarken, LR daha az eğitim verisiyle bile iyi çalışabilir.
* NN, LR'ye kıyasla yavaş olacaktır.
* Ortalama doğruluk, sinir ağlarında her zaman daha iyi olacaktır

**3.4.2.2. Lojistik Regresyon**

Doğrusal regresyon gibi, Lojistik regresyon da sınıflandırma algoritmaları ile başlamak için doğru algoritmadır. 'Regresyon' adı olsa da, bu bir regresyon modeli değil, bir sınıflandırma modelidir. İkili çıktı modelini çerçevelemek için lojistik bir işlev kullanır. Lojistik regresyonun çıktısı bir olasılık (0≤x≤1) olacaktır ve çıktı olarak ikili 0 veya 1'i tahmin etmek için kullanılabilir (x < 0.5, çıktı = 0, aksi takdirde çıktı = 1)

Avantajları:

* Kolay, hızlı ve basit sınıflandırma yöntemi.
* θ parametreler, bağımsız değişkenlerin bağımlı değişken üzerindeki öneminin yönünü ve yoğunluğunu açıklar.
* Çok sınıflı sınıflandırmalar için de kullanılabilir.
* Kayıp işlevi her zaman dışbükeydir

Dezavantajları:

* Doğrusal olmayan sınıflandırma problemlerine uygulanamaz.
* Uygun özellik seçimi gereklidir.
* Eşdoğrusallık ve aykırı değerler LR modelinin doğruluğunu bozar.

LR İçin Varsayımlar

Lojistik regresyon varsayımları, doğrusal regresyon modeline benzer.

Lojistik Regresyon vs SVM

* SVM doğrusal olmayan çözümleri işleyebilirken lojistik regresyon yalnızca doğrusal çözümleri işleyebilir.
* Doğrusal SVM, maksimum marj çözümü elde ettiği için aykırı değerleri daha iyi yönetir.
* SVM'deki hinge loss, LR'deki günlük kaybından daha iyi performans gösterir.

Lojistik Regresyon vs Karar Ağaçları (Decision Tree)

* Karar ağacı, eşdoğrusallığı LR'den daha iyi yönetir.
* Karar ağaçları özelliklerin önemini çıkaramaz, ancak LR yapabilir. - Karar ağaçları, kategorik değerler için LR'den daha iyidir.

Lojistik Regresyon vs Sinir Ağı (NN)

* NN, LR'nin destekleyemediği doğrusal olmayan çözümleri destekleyebilir.
* LR, dışbükey kayıp işlevine sahiptir, bu nedenle yerel bir minimumda asılı kalmaz, oysa NN askıda kalabilir.
* Eğitim verileri daha az ve özellikler büyük olduğunda LR, NN'den daha iyi performans gösterirken, NN'nin büyük eğitim verilerine ihtiyacı vardır.

Lojistik Regresyon vs Naive Bayes

* Naive Bayes üretken bir modeldir, LR ise ayırt edici bir modeldir.
* Naive Bayes küçük veri kümeleri ile iyi çalışır, oysa LR düzenlileştirme benzer performans sağlayabilir.
* Naive Bayes tüm özelliklerin bağımsız olmasını beklediğinden, LR eşdoğrusallık konusunda Naive Bayes ten daha iyi performans gösterir.

Lojistik Regresyon vs KNN

* KNN, LR'nin parametrik bir model olduğu parametrik olmayan bir modeldir.
* KNN, Lojistik Regresyondan nispeten yavaştır.
* KNN, LR'nin yalnızca doğrusal çözümleri desteklediği doğrusal olmayan çözümleri destekler.
* LR, (tahmini hakkında) güven düzeyini türetebilirken, KNN yalnızca etiketleri çıkarabilir.

***3.4.2.*3**. Doğrusal Regresyon K-En Yakın Komşular (KNN)

K-en yakın komşular, sınıflandırma ve regresyon için kullanılan parametrik olmayan bir yöntemdir. Kullanılan en kolay makine öğrenimi tekniklerinden biridir. Yerel yaklaşımla tembel bir öğrenme modelidir.

Avantajları

* Kolay ve basit makine öğrenimi modeli.
* Ayarlanacak az sayıda hiperparametre.

Dezavantajları

* k değeri akıllıca seçilmelidir.
* Örnek boyutu büyükse, çalışma süresi boyunca yüksek hesaplama maliyeti.
* Özellikler arasında adil muamele için uygun ölçeklendirme sağlanmalıdır.

Varsayımlar

* Giriş alanı hakkında net bir şart olmalıdır.
* Uygun şekilde orta düzeyde örneklem büyüklüğü (yer ve zaman kısıtlamaları nedeniyle).
* Eşdoğrusallık ve aykırı değerler eğitimden önce ele alınmalıdır.

KNN vs Naive Bayes

* Naive bayes, KNN'nin gerçek zamanlı uygulaması nedeniyle KNN'den çok daha hızlıdır.
* Naive bayes parametrik iken KNN parametrik değildir.

KNN vs Doğrusal Regresyon

* Veriler yüksek SNR'ye sahip olduğunda KNN, doğrusal regresyondan daha iyidir.

KNN vs SVM

* SVM, aykırı değerlere KNN'den daha iyi bakıyor.
* SVM, büyük özellikler ve daha az eğitim verisi olduğunda KNN'den daha iyi performans gösterir.

KNN vs Sinir Ağları (NN)

* Sinir ağları, yeterli doğruluğu elde etmek için KNN'ye kıyasla daha büyük eğitim verilerine ihtiyaç duyar.
* NN, KNN'ye kıyasla çok fazla hiperparametre ayarına ihtiyaç duyar.

**3.4.2.4. Karar Ağacı (Decision Tree)**

Karar ağacı, regresyon ve sınıflandırma problemlerini çözmek için kullanılan ağaç tabanlı bir algoritmadır. Çıktı türetmek için homojen bir olasılık dağıtılmış kök düğümünden oldukça heterojen yaprak düğümlerine dallanmış ters bir ağaç çerçevelenir. Regresyon ağaçları, sürekli değerlere sahip bağımlı değişken için ve ayrık değerli bağımlı değişken için sınıflandırma ağaçları kullanılır.

Avantajları

* Veriler üzerinde ön işleme gerek yoktur.
* Verilerin dağıtımı konusunda varsayım yok.
* Eşdoğrusallığı verimli bir şekilde yönetir.
* Karar ağaçları, tahmin üzerinde anlaşılır bir açıklama sağlayabilir.

Dezavantajları

* Yüksek saflık elde etmek için ağacı inşa etmeye devam edersek, modele overfitting uygulamış olabiliriz. Bu sorunu çözmek için karar ağacı budama kullanılabilir.
* Aykırı değerlere eğilimli.
* Karmaşık veri kümelerini eğitirken ağaç çok karmaşık hale gelebilir. - Sürekli değişkenleri işlerken değerli bilgileri kaybeder.

Karar Ağaçları vs Rastgele Orman

* Rastgele Orman, karar ağaçlarının bir koleksiyonudur ve ormanın ortalama / çoğunluk oyu tahmin edilen çıktı olarak seçilir.
* Rastgele Orman modeli, Karar ağacına göre fazla uyum göstermeye daha az meyilli olacak ve daha genel bir çözüm sunacaktır.
* Random Forest, karar ağaçlarından daha sağlam ve doğrudur.

Karar Ağaçları vs KNN

* Her ikisi de parametrik olmayan yöntemlerdir.
* Karar ağacı, otomatik özellik etkileşimini desteklerken KNN olamaz.
* KNN'nin pahalı gerçek zamanlı uygulaması nedeniyle karar ağacı daha hızlıdır.

Karar Ağaçları vs Naive Bayes

* Karar ağacı ayırt edici bir modeldir, Naive bayes ise üretici bir modeldir.
* Karar ağaçları daha esnek ve kolaydır.
* Karar ağacı budama, eğitim verilerindeki bazı temel değerleri ihmal edebilir ve bu da bir atış için doğruluğa yol açabilir.

Karar Ağaçları vs Sinir Ağları (NN)

* Her ikisi de doğrusal olmayan çözümler bulur ve bağımsız değişkenler arasında etkileşime sahiptir.
* Eğitim verilerinde büyük kategorik değerler kümesi olduğunda karar ağaçları daha iyidir.
* Senaryo, kararla ilgili bir açıklama talep ettiğinde, karar ağaçları NN'den daha iyidir.
* Yeterli eğitim verisi olduğunda NN, karar ağacından daha iyi performans gösterir.

Karar Ağaçları vs SVM

* SVM, doğrusal olmayan problemleri çözmek için çekirdek numarası kullanırken, karar ağaçları problemi çözmek için girdi uzayında hiper dikdörtgenler türetir.
* Karar ağaçları kategorik veriler için daha iyidir ve eşdoğrusallığı SVM'den daha iyi ele alır.

**3.4.2.5. Destek Vektör Makinesi(SVM)**

Destek Vektör makinesi, hem sınıflandırma hem de regresyon için kullanılabilen bir ML tekniği türüdür. Doğrusal ve doğrusal olmayan sorunları desteklemek için başlıca iki çeşidi vardır. Doğrusal SVM'nin çekirdeği yoktur ve soruna minimum marjlı doğrusal çözüm bulur. Çekirdekli SVM, çözüm doğrusal olarak ayrılabilir olmadığında kullanılır.

Avantajları

* SVM, karmaşık çözümleri çözmek için çekirdek numarası kullanır.
* SVM, küresel minimuma her zaman ulaşılabilen dışbükey bir optimizasyon işlevi kullanır.
* Hinge(menteşe) kaybı daha yüksek doğruluk sağlar.
* Aykırı değerler, yumuşak kenar boşluğu sabiti C kullanılarak iyi bir şekilde ele alınabilir.

Dezavantajları

* Hinge(menteşe) kaybı seyrekliğe yol açar.
* Hiper parametreler ve çekirdekler, yeterli doğruluk için dikkatlice ayarlanmalıdır.
* Daha büyük veri kümeleri için daha uzun eğitim süresi.

SVM vs Karar Ağaçları

* Random Forest çok sınıflı sınıflandırmayı desteklerken, SVM'nin bunun için birden fazla modele ihtiyacı vardır.
* Rastgele Orman, tahmin üzerinde bir olasılık verebilir, oysa SVM veremez.
* Random Forest, kategorik verileri SVM'den daha iyi ele alır.

SVM vs Naive Bayes

* Her ikisi de düşük miktarda eğitim verisi ve büyük özelliklerle daha iyi performans gösterir.
* Özellikler karşılıklı olarak bağımlıysa SVM, Naive Bayes'ten daha iyi performans gösterir.
* SVM ayırt edici bir modeldir, NB ise üretken modeldir.

SVM vs Yapay Sinir Ağları (NN)

* SVM'nin dışbükey bir optimizasyon işlevi vardır, oysa NN yerel minimumda asılı olabilir.
* SVM, sınırlı eğitim verisi ve birçok özellik olduğunda NN'den daha iyi performans gösterebilir. NN, yeterli doğruluk için büyük eğitim verilerine ihtiyaç duyar.
* Çok sınıflı sınıflandırma, SVM için birden fazla model gerektirirken, NN bunu tek bir modelle yapabilir.

**3.4.2.6. Rastgele Orman(Random Forest)**

Random Forest, daha güçlü bir model elde etmek için çoklu karar ağaçlarının birleştirildiği bir topluluk modelidir. Türetilen model daha sağlam, doğru olacak ve aşırı uyumu kurucu modellerden daha iyi idare edecektir.

Avantajları

* Doğru ve güçlü model.
* Aşırı uydurmayı verimli bir şekilde ele alır.
* Örtük özellik seçimini destekler ve özellik önemini türetir.

Dezavantajları

* Orman büyüdüğünde hesaplama açısından karmaşık ve daha yavaş. - Tahmin üzerinde iyi tanımlayıcı bir model değil.

Rastgele Orman vs Naive Bayes

* Rastgele Orman karmaşık ve büyük bir modelken, Naive Bayes nispeten daha küçük bir modeldir.
* Naive Bayes, küçük eğitim verileriyle daha iyi performans gösterirken, RF'nin daha büyük eğitim verileri setine ihtiyacı vardır.

Rastgele Orman vs Yapay Sinir Ağları(NN)

* Her ikisi de çok güçlü ve yüksek doğruluklu algoritmalardır.
* Her ikisinin de dahili olarak özellik etkileşimleri vardır ve daha az açıklanabilir.
* Random Forest özellik ölçeklendirmeye ihtiyaç duymazken, NN özelliklerin ölçeklendirilmesine ihtiyaç duyar.
* Her iki modelin toplu versiyonu güçlü olacak.

**3.4.2.7 Naive Bayes**

Naive bayes, sınıflandırma problemleri için kullanılan üretken bir olasılık modelidir. Özellik setinin çok büyük olduğu metin sınıflandırmaları için kullanılan ana modeldir. Duygu analizi, spam filtreleme vb. İçin yaygın olarak kullanılmaktadır.

**3.4.3. Müşteri Yorumları Hissiyat Analizi Uygulaması**

Veri Seti : https://www.kaggle.com/burhanbilenn/turkish-customer-reviews-for-binary-classification

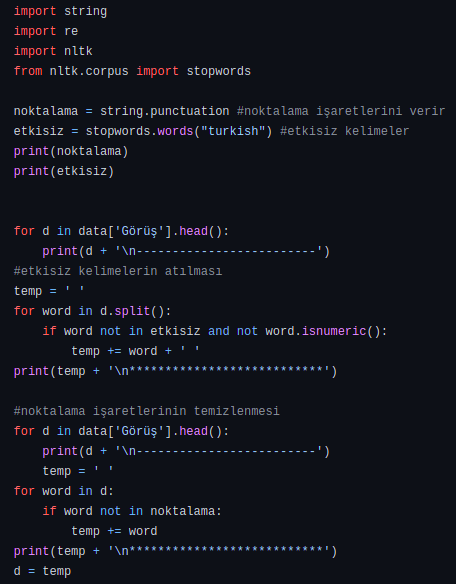
**3.4.3.1. Veri Seti Ön İşleme Adımları**

onisleme.py

İlk olarak veri setimizi python üzerinde tanımplayıp NaN değerleri kaldırdım.



Ardından stop words(etkisiz kelimeleri) ve noktalama işaretlerini bulup kaldırdım.



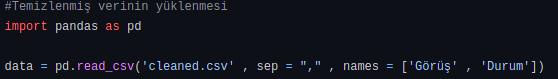
Ardından temizlemiş olduğum veriyi “cleaned” adını verdiğim .csv uzantılı dosyaya yazdırdım.



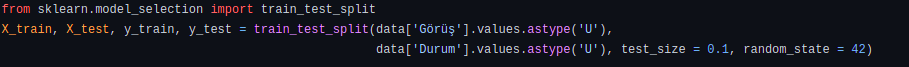
**3.4.3.2. Uygulama Bölümü**

app.py

İlk olarak temizlenmiş veriyi yükledim.



Ardından veriyi train ve test kümelerine ayırdım.

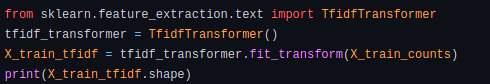


Train kümesindeki vektörlerin sayma vektörlerini çıkardım.

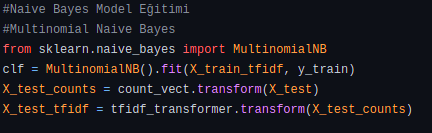


Train kümesindeki cümlelerin TF\*IDF vektörlerini sayma vektörlerinden oluşturdum

TF\*IDF bir doküman uzayi içerisinde geçen bir kelimenin herhangi bir doküman içerisinde ne kadar önemli olduğunu belirtmek için tasarlanmış istatiksel bir ölçüttür.



Sınıflandırmak için Nmultinominal Naive Bayes modelini kullandım.



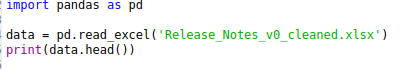
Accuracy Puanımız ise 0.9199057714958775.

Proje Kaynak Dosyaları: https://github.com/icanardahan/Hissiyat-Analizi-NaiveBayes.git

**3.4.4. Relase Notları Veri Seti üzerinde Makine Öğrenmesi ve Derin Öğrenme Algoritmalarınının Uygulanması**

**3.4.4.1. Naive Bayes Algoritmasının Uygulanması**

İlk olarak veri setini yüklüyoruz.

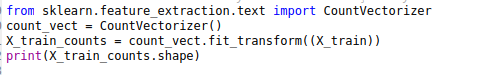


Ardından Train ve Test Kümelerine Ayırıyoruz.



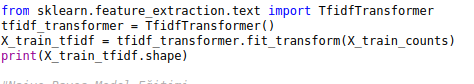
Bu işlemlerden sonra sayma vektörümüzü oluşturuyoruz.

İlk olarak Train kümesindeki cümlelerin sayma vektörlerini çıkarıyoruz.

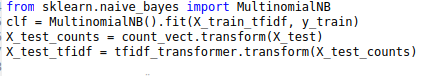


Train kümesindeki cümlelerin TF\*IDF vektörlerini sayma vektörlerinden oluşturuyoruz.

TF-IDF, bir doküman uzayi içerisinde geçen bir kelimenin herhangi bir doküman içerisinde ne kadar önemli olduğunu belirtmek için tasarlanmış istatiksel bir ölçüttür.



Multinomial Naive Bayes Modeli Oluşturma:



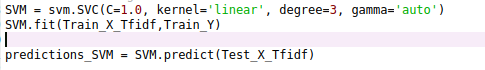
Test Sonuçları:



**Accuracy Değeri:** 0.6962025316455697

**3.4.4.2. SVM Algoritmasının Uygulanması**

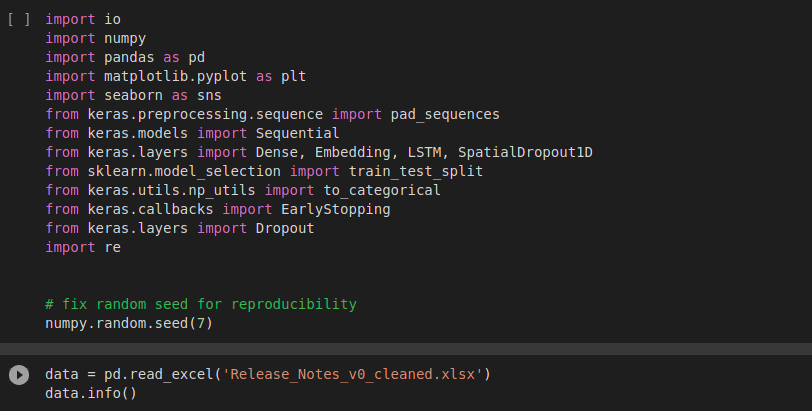
Yukarıda anlatılan Naive Bayes uygulamasıyla çoğu adım bire bir aynı şekilde ilerliyor. Sadece değiştireceğimiz aşama “Multinomial Naive Bayes Modeli Oluşturma” kısmıdır.Bu kısım yerine SVM algoritmamız için gerekli SVM modelini oluşturacağız.



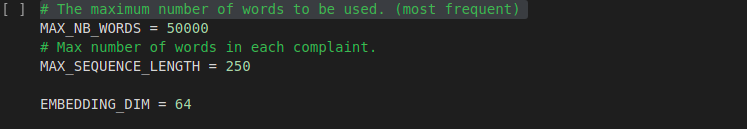
**Accuracy Değeri:** 0.7257383966244726

**3.4.4.3. LSTM Algoritmasının Uygulanması**

Gerekli kütüphanelerin yüklenmesi ve veri setinin okunması:

****

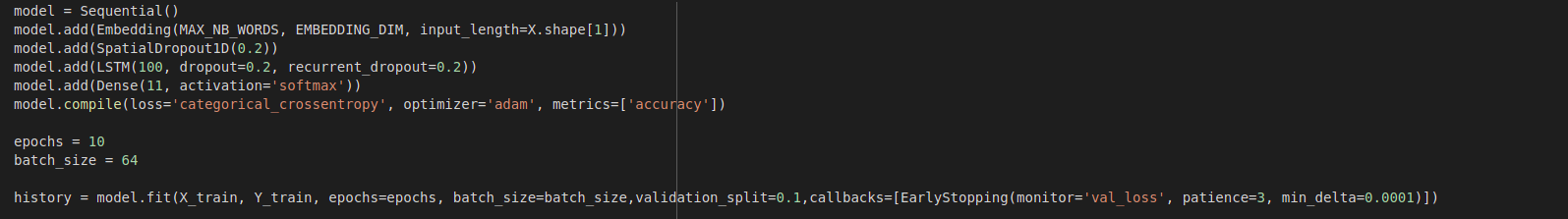
Bir sonraki adımda ise kullanılacak maksimum kelime sayısı ve bir cümlede geçebilecek maksimum kelime sayısını ayarlıyoruz.



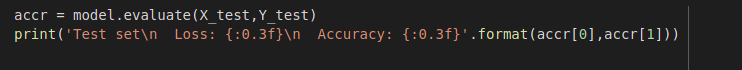
Veri setinin test ve train kümelerine ayrılması:



Ardından LSTM modelimizi oluşturuyoruz.



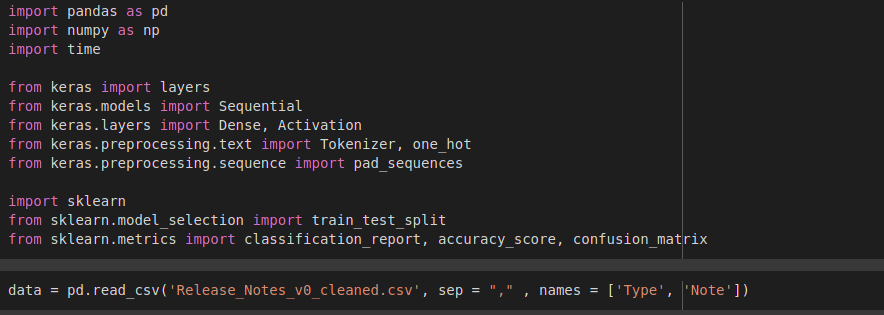
Modelin başarı oranı:



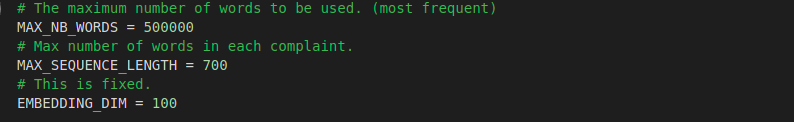
**Accuracy Değeri:** 0.405

**3.4.4.4. CNN Algoritmasının Uygulanması**

Gerekli Küttüphaneler ve Veri Setinin yüklenmesi;

****

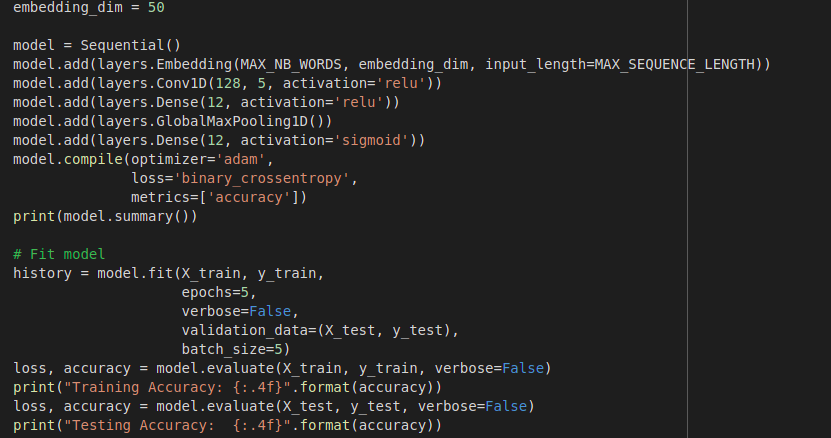
Kullanılacak maksimum kelime sayısı ve bir cümlede geçebilecek maksimum kelime sayısının ayarlanması.

****

Train ve test kümelerinin ayrılması

****

**CNN modelinin oluşturulması ve başarı oranı**

****

**Accuracy Değeri :** 0.443

**4.ARAŞTIRMA SONUÇLARI VE TARTIŞMA**

4.1. Projede Kullanılan Yöntem Araştırması

Proje araştırmasında github’ta en çok tercih edilen programlama dilleri ve proje araştırmasına hakim olundu.

Github’ta 4 proje çalışmasının release notları excel de “tablo ve data” sayfaları olarak veri çekme işlemleri yapıldı. Proje için gerekli olan release notları elde edildi.

5. SONUÇLAR VE ÖNERİLER

## **5.1 Sonuçlar**

Proje için gerekli olan release notları not edildi ve oluşturuldu

**6.KAYNAKLAR**

<https://www.slideshare.net/saho10/datk-sistemler-programlama>

<https://www.kizgibikodla.com/news/vcs-git-github-bitbucket-nedir-github-kullanimi/>

<http://www.sistemvenetworkmuhendisi.com/dagitik-mimari-nedir-nasil-calisir/#:~:text=Da%C4%9F%C4%B1t%C4%B1k%20sistem%2C%20birden%20fazla%20bilgisayar,ve%20kullan%C4%B1c%C4%B1%20a%C3%A7%C4%B1s%C4%B1ndan%20saydaml%C4%B1k%20sa%C4%9Flan%C4%B1r>.

<https://www.chip.com.tr/haber/acik-kaynak-kod-nedir-size-de-gerekli-mi_38478.html>

<https://stackoverflow.com/questions/33551505/what-exactly-is-a-release-in-github#:~:text=From%20Official%20GitHub%3A,release%20notes%20describing%20your%20changes>.

<https://www.tasarimkodlama.com/genel/teknoloji/acik-kaynak-nedir/>

<https://github.com/vigo/git-puf-noktalari/blob/master/bolum-01/repository-nedir.md>

https://github.com/obsproject/obs-studio

<https://slack.com/intl/en-tr/release-notes/windows>

https://pypi.org/project/porter2stemmer/

https://porter2-stemmer.readthedocs.io/en/latest/

https://ichi.pro/tr/nlp-ye-giris-bolum-2-serbestlestirme-ve-kok-belirleme-arasindaki-fark-61064739575228

http://hunterheidenreich.com/blog/stemming-lemmatization-what/

https://pypi.org/project/porter/

https://github.com/shunfan/porter

https://packaging.python.org/guides/analyzing-pypi-package-downloads/

https://texthero.org/

https://pypi.org/project/snowballstemmer/

http://buyukveri.firat.edu.tr/2018/06/04/metin-on-isleme-adimlari-icin-keras-tokenizer-sinifi-kullanimi/

https://www.nltk.org/\_modules/nltk/stem/lancaster.html#LancasterStemmer.parseRules

https://www.nltk.org/api/nltk.stem.html#nltk.stem.lancaster.LancasterStemmer

https://www.nltk.org/\_modules/nltk/stem/snowball.html#ArabicStemmerhttps://www.nltk.org/\_modules/nltk/stem/snowball.html#ArabicStemmer

https://tr.delavaio.com/105256-python-text-processing-nltk-and-XDCYTV

https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8