**Python'da Majority Vote (Topluluk Oylaması) ile Müşteri Kayıp(Churn) Analizi**

Merhabalar, Bugünkü yazımda sizlere python'da nasıl Müşteri kayıp analizi yapacağınızdan ve Majority Vote(Topluluk oylaması) sisteminden bahsedeceğim. Yazımızın içeriği şu şekilde olacak.

* Müşteri Kaybı (Churn) Nedir?
* Churn'e yapabileceğimiz farklı yaklaşımlar
* Keras Yaklaşımı nedir?
* Risk Tabanlı Makine öğrenmesi yaklaşımı nedir?
* Geleneksel Gözetimli Makine öğrenmesi yaklaşımı nedir?
* Majority Vote sistemi nedir?

**Tam Faktoring Verisetinde Churn Analizi**

* Projenin Amacı
* Etiketsiz Churn Verisini Analiz Edip Etiketleme
* Veri Özetleme algoritması
* Boş Veri Yaklaşımı
* Temizlendikten Sonra Veriseti Profili
* Risk Tabanlı/Geleneksel/Keras ile Harmanlanan Topluluk Oylamasının Projeye Entegrasyonu
* Geleneksel Makine Öğrenmesi Algoritması
* Risk Tabanlı Makine Öğrenmesi Algoritması
* Keras Bazlı Derin Öğrenme Algoritması
* Topluluk Oylaması / Majority Vote Algoritması
* Genel Sonuçlar

**Müşteri Kayıp (Churn) Nedir?**

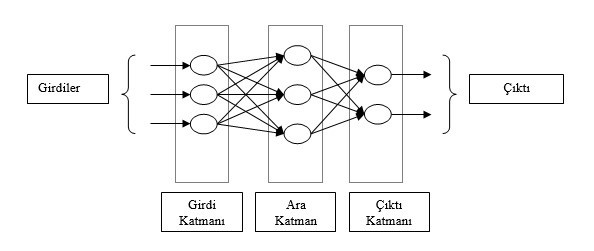
* Çoğunlukla Telekom, Sigortacılık, Bankacılık gibi sektörlerde sürekli müşteri'nin kaybını ifade eder. Örnek olarak Telekom sektöründe X Operatöründe 10 senelik müşteri'nin Y Operatörü'ne geçmesi X Operatörü için Müşteri kaybıdır.
* Müşteri Kayıp Analizi ise bu gidecek olan müşteri'nin önceden yaptığı hareketlere göre izlenip, başka bir şirket/firma/operatöre geçeceğini tahminleyip, Müşteriye özel kampanya/paket/indirim ile müşteriyi kaybetmemek amaçlanır.

**Churn'e yapabileceğimiz farklı yaklaşımlar**

* Churn analizinde birçok yaklaşım bulunmakta bunların en çok kullanılanları gözetimli makine öğrenmesi ile önceden bırakan müşteriler'in verileri ve sürekli temiz müşterilerin verileri model'e öğretilir. Ve yeni müşteriler için sürekli sorgulama yapılır. Churn olma eğilimi olan müşteri Churn diye etiketlenir.
* Derin öğrenme yaklaşımında da aynı şekilde veriler öğretilir. Ve yeni müşteriler için öğrenilen verilerden 0-1 arasında bir risk skoru çıktısı alınır. Ardından bu model için bir eşik değeri belirlenir. Mesela 0.60 Eğer kişi için atanan risk skoru 0.60'ın üzerindeyse kişi Churn edecek olarak etiketlenir. Altındaysa sürekli olarak işaretlenebilir.
* Son yaklaşım ise kişi bazlı risk skorlama yaklaşımıdır. Burada çalışma stili geleneksel gözetimli öğrenme ile aynıdır. Ama risk bazlı skorlama derin öğrenme gibi 0-1 arasında bir değer atayabilir. Bu değeride eşik değeri yardımı ile etiketleyebiliriz. Burada eşik değerinin değişimi model'in sonuçlarını etkilediği gibi şirket bazında da kar maksimizasyonu'nun sağlanmasında önemli rol oynar.

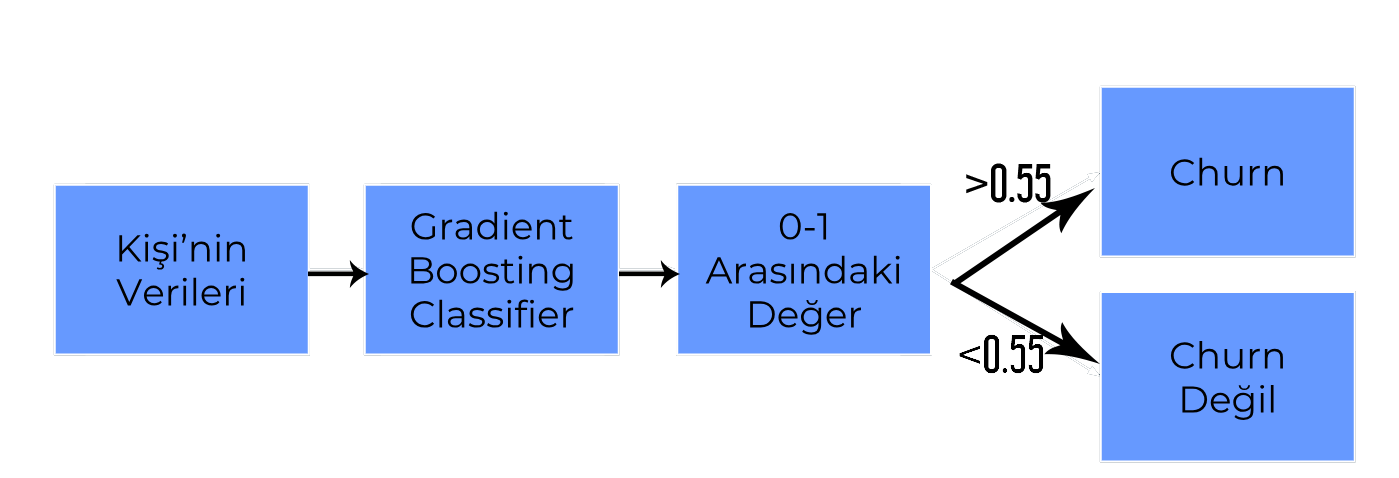
**Keras Yaklaşımı nedir?**

* Derin Öğrenme (Yapay Sinir Ağları) Yaklaşımında ise Amaç, ağa gösterilen örnekler için doğru çıktıları üretecek ağırlık değerlerini bulmaktır. Ağ’ın doğru ağırlıkları bulunduktan (Model eğitildikten) sonra ise yapılan işlemin Olasılık Bazlı Makine Öğrenmesinden farkı yoktur. 0 ile 1 arasında bir değer atanır ve yine bizim belirlediğimiz eşik değerine göre (Genelde 0.5) bu değerler 0 veya 1’e döndürülür. Çalışma mantığı ise aşağıdaki gibidir. Bu yöntemde de aynı şekilde eşik değeri’nin değişimi şirket’in karlılığını etkiler.
* Buradaki kurduğumuz yapay sinir ağı şu şekilde birşey olacaktır :



**Risk Tabanlı Makine öğrenmesi yaklaşımı nedir?**

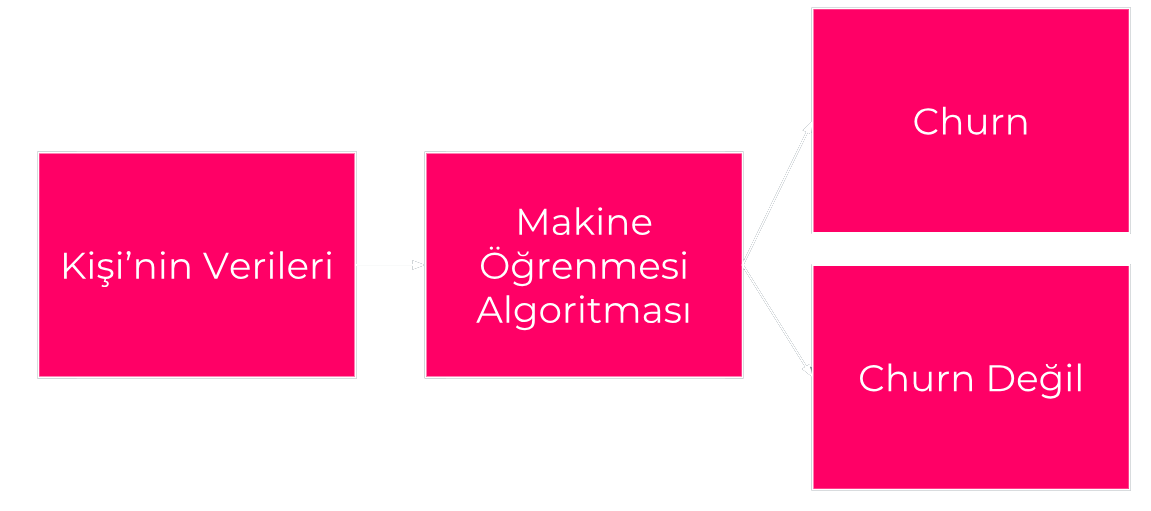
* Olasılık Bazlı makine öğrenmesi yaklaşımında ise verisetine yapılan işlemler aynıdır. Bu yaklaşımda her müşteri için 0 ile 1 arasında bir risk skoru döndürülür. Ve belirlenen eşik değeri vasıtasıyla bu değer 1 veya 0 olarak etiketlenir. Eşik değeri şirket’in kar maksimizasyonunu sağlayacak şekilde belirlenir. Çalışma mantığı ise aşağıdaki gibidir.
* Şemadaki Gradient Boosting Classifier popüler makine öğrenmesi algoritmalarından biridir.



**Geleneksel Gözetimli Makine öğrenmesi yaklaşımı nedir?**

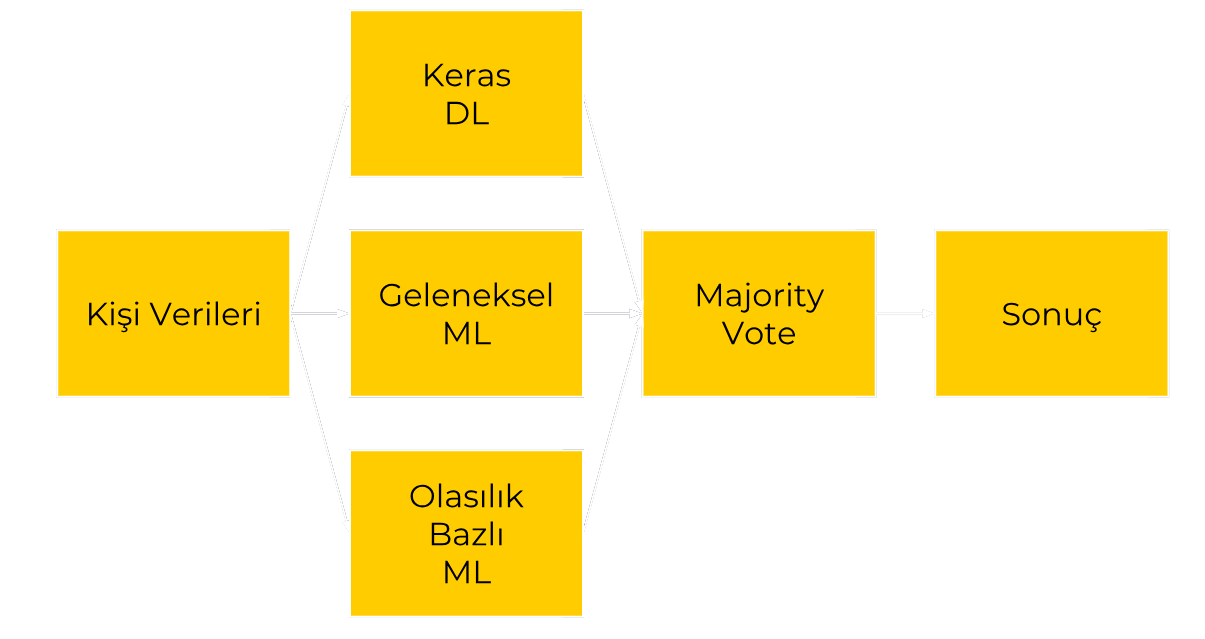
[**(Makine Öğrenmesi Nedir? Bilmiyorsanız Tıklayın)**](https://www.linkedin.com/pulse/makine-öğrenmesi-nedir-pythonda-ml-modeli-nasıl-gerçek-gündoğmuş/)

* Geleneksel Gözetimli Makine Öğrenmesi yaklaşımında Elimizdeki Veriseti Eğitim ve Test olmak üzere 2’ye bölünür. Eğitim setindeki veriler hedef değişken ile beraber öğrenilir. Ardından test verisetindeki değerlerde eğitim verisetindekilere benzetilerek tahminlemeler yapılır. Ve bu tahminlemeler gerçek değerlerle karşılaştırarak model’in doğruluğuna bakılır.



**Majority Vote sistemi nedir?**

* Majority Vote(Topluluk Oylaması)’un mantığı sisteme dahil olan elemanlar bir oy verir. Ve en çok oy’u alan çıkarım doğru olarak kabul edilir.
* Örnek vermek gerekirse, Seçimi düşünebilirsiniz, milyonlarca insan oy veriyor. Ve en çok oyu alan kişi kazanıyor. Veya 3 adet disiplin kurulu üyesi öğrencinin suçluluğu üzerine konuşuyorlar. 2 üye suçlu 1 üye suçsuz diyorsa öğrenci suçlu olarak kabul edilir. bu problemde de aynı durum söz konusu, 3 adet farklı yaklaşım var. Bu 3 farklı yaklaşım kişiye ait verilere bakarak kişi için churn veya değil şeklinde çıkarım yapıyor ve oy veriyor. Oylar'ın toplamına bakıldığında en çok oyu alan çıkarım doğru olarak kabul ediliyor.
* Bizim yapacağımız örnek projede Majority vote sisteminin çalışma mantığı 3 farklı yaklaşım (Derin Öğrenme, Risk Tabanlı Skorlama, Geleneksel Makine Öğrenmesi) kişi için çıkarım yapar ve oy verir. Verilen oylara bakılarak kişi'nin churn edip etmeyeceği belirlenir.



Evet Majority Vote sisteminin nasıl çalıştığını da anladıysak artık Tam Faktoring'te yaptığımız Detective Churn Projesinden bahsedeyim.

**Tam Faktoring Verisetinde Churn analizi projesi**

**Projenin Amacı**

Projemizdeki amacımız belli bir işlem sayısından sonra çek bozdurmayı bırakan müşterileri tespit etmekti. Burada normal Müşteri Kayıp Analizi yerine farklı bir bakış açısı ile bu probleme farklı açılardan bakmaya karar verdik. Bu Açılar :

* Geleneksel Yaklaşım (Geleneksel ML)
* Şirketin Kâr Maksimizasyonunu Düşünen Yaklaşım (Risk Bazlı ML)
* Güncel Teknoloji Yaklaşımı (Derin Öğrenme)

Şeklindeydi.

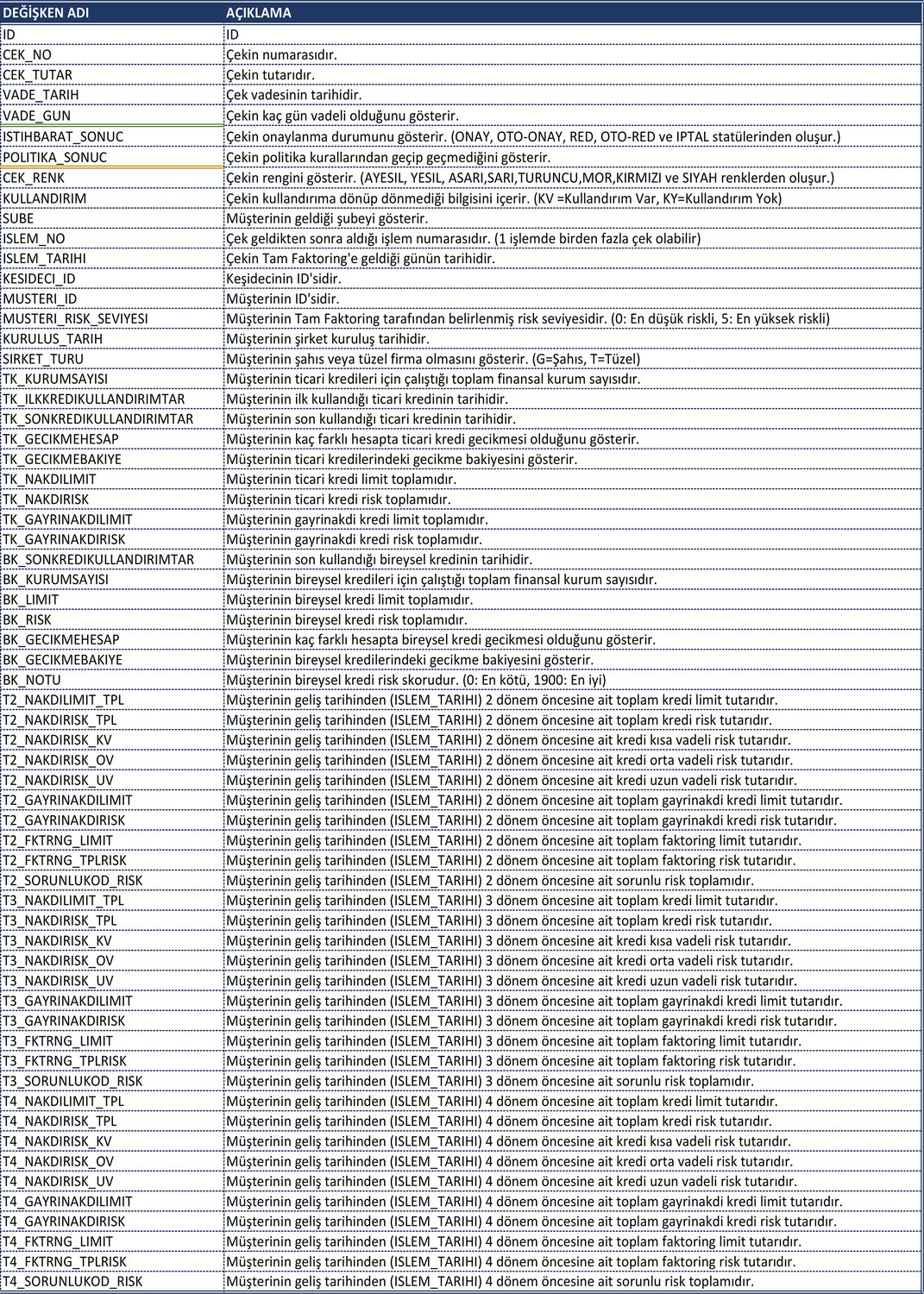
3 Farklı Yöntem kullanıp bu yöntemler üzerinden bir Topluluk Oylaması(Majority Vote) oluşturmaya karar verdik. Burada kullandığımız yöntemler ise ;

* Derin Öğrenme
* Geleneksel Makine Öğrenmesi
* Olası Risk Skorları Üzerinden Belirlenen Cut-Off ile Etiketleme Yaklaşımı

Bu 3 bakış açısı üzerinden kişiyi Churn edecek veya etmeyecek şeklinde tahminledik.

**Veri Analizi**

Verisetimiz **2017 Ocak - 2018 Kasım** arasındaki Tam Faktoring'e başvuran müşteriler'in çek bilgilerini içeriyordu. **63 Değişken** ve **800.000** Satırlık Bir veriydi. Değişkenlerimiz ise :



Burada en büyük sorun veride label(hedef) değişken olmamasıydı. Bizim kuracağımız bütün modeller ise Hedef değişken üzerinden kurgulandığı için, Verideki müşterileri etiketlemeye karar verdim.

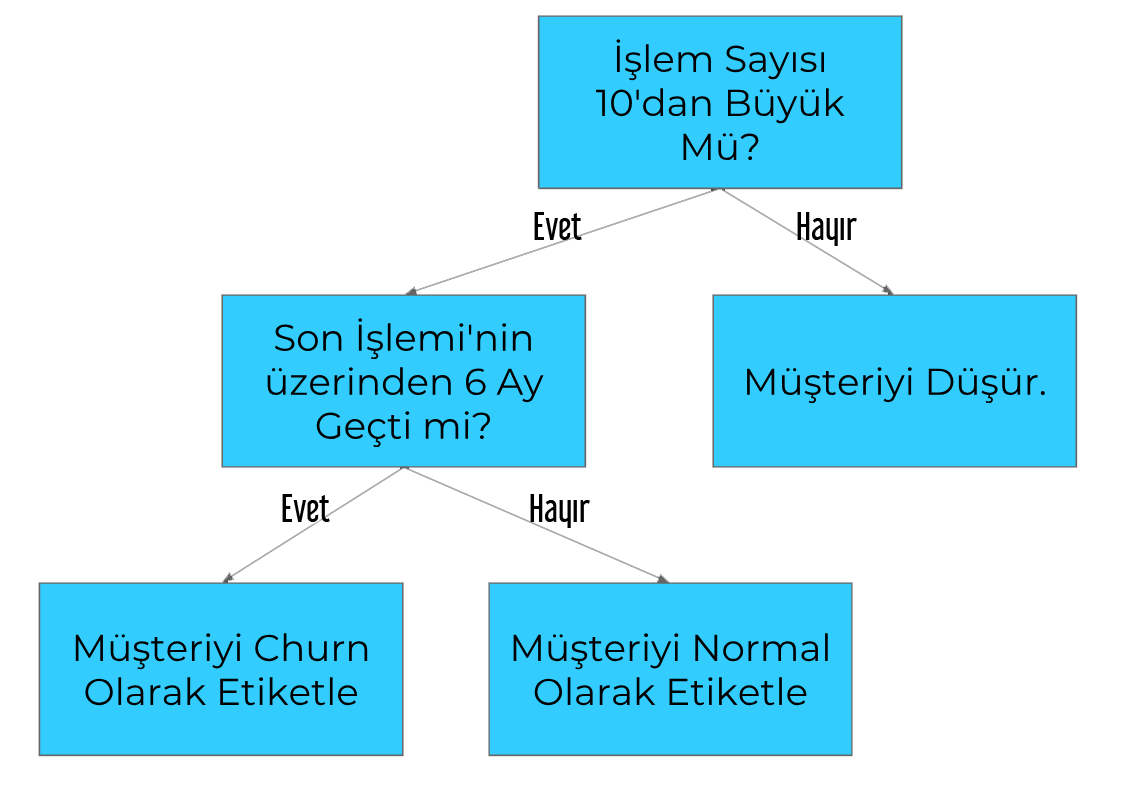
**Etiketsiz Churn Verisini Analiz Edip Etiketleme**

Öncelikle Sürekli müşterileri almak için veriyi **10 işlem üstü müşterileri al** şeklinde filtreledim. Buradaki amaç'ta az verisi olan müşterileri(1 işlem-2 işlem) verisetime alırsam doğru bir churn modeli kuramayabilirdim. Buradaki 10 Sayısıda normalde işlem ortalaması 4 ben bunun 2.5 katını alarak, tam anlamıyla sürekli müşterileri elde etmek istedim.

Proje'nin en önemli kısmı da buydu aslında çünkü burada yalnış yerden etiketlersem skorlarım ve kurduğum model'in bir anlamı olmayabilirdi. Burada hem veriyi analiz ederek ortalama işlem aralığı gününü hemde tam faktoring'in hitap ettiği çek kullanan kitleye sorduğumuz sorular ile aldığımız cevapları harmanladık ve burada son işlemi üzerinden 6 ay geçen insanları Churn geçmeyenler Churn değil şeklinde işaretledik.

[(Bu Kısımların Kodlarına Tıklayarak Erişebilirsiniz.)](https://github.com/unforgiventr/Detective-Churn)

Bu kısımda yaptıklarımızı kısaca şemalaştırmak gerekirse :



**Veri Özetleme Algoritması**

Bu işlemleri yapınca yapımıza uyan 19.990 Müşteri bulunuyordu. Bu müşterilerin bütün işlemleri toplamda 660.000 Satırdan oluşuyordu. Buradaki yeni sorunumuz ise müşteriyi işlemleri bazından değerlendirirsek yalnış sonuçlar elde edeceğimiz için bu işlemleri özetlemek ve müşteriyi son işlemi bazında değerlendirmekti. Bunun içinde karşımıza yeni sorular geldi :

* Müşteriyi Bütün işlemleri bazında mı Yoksa belli bir son işlem bazında mı değerlendirecektik? (Son 5)
* Hangi değişkenler bizim için işlevsiz olacaktı?

Buradaki soruların cevabını bulabilmek için yine tecrübelilere sorduk, daha önceki çalışmalara göz attık. Ve buradaki aksiyonumuzu son 5 işlem olarak belirledik sebebi ise aynı çeki birkaç kez bozdurmayı deneyenler, bunlarda hata yapmamak için ve aynı zamanda müşteri yaşam döngüsünde son hareketlerde alınan bir cevap/faiz oranı etkilediği için son 5 işlemi almaya karar verdik. Burada işlevsiz olduğunu düşündüğümüz içinde ID'ler, Tarih bilgilerini kullanmamaya karar verdik.

Ardından Özetleme algoritmasını veriye uygulayarak, 660.000 Satırlık veriyi 19.990 Satıra indirgedik. Buradaki yaklaşımımızda şu şekildeydi :



Burada kalan değişkenler içinde aynı işlemleri uyguladık, Kategorik veriler için Son 5 işlemdeki en yüksek frekans'a sahip olaya, Sürekliler içinde Son 5 işlem'in ortalamasına baktık. Veriler'in temizlenmesinde son olarak Boş veriler ile ilgili yapılacak bir iş kalmıştı.

**Boş Veri Yaklaşımı**

Verisetinde içerisinde boş hücre bulunan birkaç değişken vardı, Bunlar :

* Kredi Kayıt Bürosu Skoru : BK\_NOTU
* Risk Raporu Değerleri 2-3-4 Döneme ait Risk Raporları

Burada BK\_NOTU Değişkenini destekleyen çok değişken olduğu ve aynı zamanda doldurma’nın bizi hataya sürükleyeceğini düşündüğümüzden bu değişkeni düşürerek kullanmadık.

Kalan risk raporu değerleri, kişinin onay veya red’i için çok önemli değerler olduğundan onları 0 ile doldurarak etkisiz hale getirmeye karar verdik.

**Temizlendikten Sonra Veriseti Profili**

Verimiz Özetlendikten ve Temizlendikten Sonra içerisinde **19.990 Satır , 55 Değişken** bulunmaktaydı.

Verimizdeki hedef değişken dağılımı **1614 Churn** ve **18.375 Churn Olmayan** şeklinde olduğu için Verisetinde küçülmeye gittik. Ve bütün modellerde Verisetini Aşırı öğrenme yaşamamak için Churn olmayan veri sayısını churn olan veri sayısı’nın **3 katı** olacak şekilde belirledik. Aynı zamanda hesaplama da hata olmaması için bütün yaklaşımlar için eğitim ve test verisi verisetinin aynı kısmından alındı.

[(Buraya Kadarki Kodlara Erişmek için Tıklayın)](https://github.com/unforgiventr/Detective-Churn)

Not : Yazının bu kısmına kadar normalde yazılarda kullandığımdan farklı olarak kod yazmadım, sebebi ise linkteki notebook'ta detaylı şekilde nasıl çalıştığını anlattım. Oradan bakabilirsiniz. Şimdi ise daha detaylı ve kodlu bir şekilde devam edeceğim.

**Risk Tabanlı/Geleneksel/Keras ile Harmanlanan Topluluk Oylamasının Projeye Entegrasyonu**

Evet burada öncelikle Geleneksel Makine Öğrenmesi algoritmasının nasıl kurulduğundan bahsetmek istiyorum.

**Geleneksel Makine Öğrenmesi Algoritması**

#Kullanacağım Kütüphaneleri Çağırıyorum.

import numpy as np

import pandas as pd

from sklearn.ensemble import GradientBoostingClassifier

from sklearn import preprocessing

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix, accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.model\_selection import train\_test\_split

import pickle

#Veriyi Tanıtıyorum

data = pd.read\_excel("newdataFullv3.xlsx")

data ​= data.dropna()

columns = ['CEK\_TUTAR',

 'VADE\_GUN',

 'SON5ISTIHBARAT\_SONUC',

 'SON5POLITIKA\_SONUC',

 'SON5CEK\_RENK\_ORTALAMASI',

 'SON5KULLANDIRIM',

 'ŞUBE',

 'SIRKET\_TURU',

 'MUSTERI\_RISK\_SEVIYESI',

 'TK\_KURUMSAYISISON5',

 'TK\_GECIKMEHESAPSON5',

 'TK\_GECIKMEBAKIYEORT',

 'TK\_NAKDILIMIT',

 'TK\_NAKDIRISK',

 'TK\_GAYRINAKDILIMIT',

 'BK\_KURUMSAYISI',

 'BK\_LIMIT',

 'BK\_RISK',

 'BK\_GECIKMEHESAP',

 'BK\_GECIKMEBAKIYE',

 'T2\_NAKDILIMIT\_TPL',

 'T2\_NAKDIRISK\_TPL',

 'T2\_NAKDIRISK\_KV',

 'T2\_GAYRINAKDILIMIT',

 'T2\_GAYRINAKDIRISK',

 'T2\_FKTRNG\_LIMIT',

 'T2\_FKTRNG\_TPLRISK',

 'T2\_NAKDIRISK\_OV',

 'T2\_NAKDIRISK\_UV',

 'T2\_SORUNLUKOD\_RISK',

 'T3\_NAKDILIMIT\_TPL',

 'T3\_NAKDIRISK\_TPL',

 'T3\_NAKDIRISK\_KV',

 'T3\_NAKDIRISK\_OV',

 'T3\_NAKDIRISK\_UV',

 'T3\_GAYRINAKDILIMIT',

 'T3\_GAYRINAKDIRISK',

 'T3\_FKTRNG\_LIMIT',

 'T3\_SORUNLUKOD\_RISK',

 'T3\_FKTRNG\_TPLRISK',

 'T4\_NAKDILIMIT\_TPL',

 'T4\_NAKDIRISK\_TPL',

 'T4\_NAKDIRISK\_KV',

 'T4\_NAKDIRISK\_OV',

 'T4\_NAKDIRISK\_UV',

 'T4\_GAYRINAKDILIMIT',

 'T4\_GAYRINAKDIRISK',

 'T4\_FKTRNG\_LIMIT',

 'T4\_SORUNLUKOD\_RISK',

 'T4\_FKTRNG\_TPLRISK',

 'Churn']

data.head()

Evet veriyi tanıtıp, kütüphaneleri çağırdıktan sonra veri tiplerine göz atarak hangi verileri nasıl kullanacağıma ve kategorik verileri belirleyerek onları makine öğrenmesine uygun hale getirmek için veri tiplerine baktım.

data.info()

Aldığım çıktı ise şu şekildeydi :

<class 'pandas.core.frame.DataFrame'>

Index: 19989 entries, 480 to SC00190565

Data columns (total 51 columns):

CEK\_TUTAR 19989 non-null float64

VADE\_GUN 19989 non-null float64

SON5ISTIHBARAT\_SONUC 19989 non-null object

SON5POLITIKA\_SONUC 19989 non-null object

SON5CEK\_RENK\_ORTALAMASI 19989 non-null object

SON5KULLANDIRIM 19989 non-null object

ŞUBE 19989 non-null object

SIRKET\_TURU 19989 non-null object

MUSTERI\_RISK\_SEVIYESI 19989 non-null int64

TK\_KURUMSAYISISON5 19989 non-null int64

TK\_GECIKMEHESAPSON5 19989 non-null int64

TK\_GECIKMEBAKIYEORT 19989 non-null float64

TK\_NAKDILIMIT 19989 non-null float64

TK\_NAKDIRISK 19989 non-null float64

TK\_GAYRINAKDILIMIT 19989 non-null float64

BK\_KURUMSAYISI 19989 non-null int64

BK\_LIMIT 19989 non-null float64

BK\_RISK 19989 non-null float64

BK\_GECIKMEHESAP 19989 non-null int64

BK\_GECIKMEBAKIYE 19989 non-null float64

T2\_NAKDILIMIT\_TPL 19989 non-null float64

T2\_NAKDIRISK\_TPL 19989 non-null float64

T2\_NAKDIRISK\_KV 19989 non-null float64

T2\_GAYRINAKDILIMIT 19989 non-null float64

T2\_GAYRINAKDIRISK 19989 non-null float64

T2\_FKTRNG\_LIMIT 19989 non-null float64

T2\_FKTRNG\_TPLRISK 19989 non-null float64

T2\_NAKDIRISK\_OV 19989 non-null float64

T2\_NAKDIRISK\_UV 19989 non-null float64

T2\_SORUNLUKOD\_RISK 19989 non-null float64

T3\_NAKDILIMIT\_TPL 19989 non-null float64

T3\_NAKDIRISK\_TPL 19989 non-null float64

T3\_NAKDIRISK\_KV 19989 non-null float64

T3\_NAKDIRISK\_OV 19989 non-null float64

T3\_NAKDIRISK\_UV 19989 non-null float64

T3\_GAYRINAKDILIMIT 19989 non-null float64

T3\_GAYRINAKDIRISK 19989 non-null float64

T3\_FKTRNG\_LIMIT 19989 non-null float64

T3\_SORUNLUKOD\_RISK 19989 non-null float64

T3\_FKTRNG\_TPLRISK 19989 non-null float64

T4\_NAKDILIMIT\_TPL 19989 non-null float64

T4\_NAKDIRISK\_TPL 19989 non-null float64

T4\_NAKDIRISK\_KV 19989 non-null float64

T4\_NAKDIRISK\_OV 19989 non-null float64

T4\_NAKDIRISK\_UV 19989 non-null float64

T4\_GAYRINAKDILIMIT 19989 non-null float64

T4\_GAYRINAKDIRISK 19989 non-null float64

T4\_FKTRNG\_LIMIT 19989 non-null float64

T4\_SORUNLUKOD\_RISK 19989 non-null float64

T4\_FKTRNG\_TPLRISK 19989 non-null float64

Churn 19989 non-null int64

dtypes: float64(39), int64(6), object(6)

memory usage: 7.9+ MB

Burada kategorik değişkenleri anladıktan sonra onları numerik hale getirmek için, label encoder kullanmaya karar verdim. Ve şöyle bir döngü yazdım.

#Labelencoder'ı çağırıyoruz.

le = preprocessing.LabelEncoder()

#Kategorik değişkenlerin isimlerini yazıyoruz.

objects = ["SON5ISTIHBARAT\_SONUC","SON5POLITIKA\_SONUC","SON5CEK\_RENK\_ORTALAMASI","SON5KULLANDIRIM","ŞUBE","SIRKET\_TURU"]

#Buradaki döngüde listenin içindeki elemanlar tek tek numerik hale döndürülüyor.

for i in objects:

    data[i] = le.fit\_transform(data[i])

Ardından başta da bahsettiğim gibi hedef değişkenin dağılımı **1614 Churn 18.375 Churn Olmayan** şeklinde olduğu için veri setini revize ettim.

print("Data'nın Bölünmeden Önceki Boyutu", len(data))

dataChurn = data[data.Churn == 1]

dataNormal = data[data.Churn == 0]

data = pd.concat([dataNormal.sample(len(dataChurn)\*3, random\_state=5,replace=True), dataChurn])

print("Data'nın Yeni Boyutu: ", len(data))

Aldığım Çıktı ise :

Data'nın Bölünmeden Önceki Boyutu 19989

Data'nın Yeni Boyutu: 6456

Ardından veriyi eğitim ve test olmak üzere 2'ye ayırdım burada veri'nin %20 'si test %80'i eğitim olarak ayrıldı.

X = data.iloc[:, data.columns != 'Churn'] #Churn dışındaki bütün değişkenler

y = data.Churn #Churn değişkeni

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 25)

Ve herşey hazır olduğuna göre artık modeli kurgulayabiliriz, burada ben örnek olarak tek modelde çalıştım. Siz bütün modelleri deneyip,ardından aralarından en iyi modeli seçebilirsiniz.

#Model'in Parametreleri

model = GradientBoostingClassifier(learning\_rate=0.15,n\_estimators=137,subsample=0.7, random\_state=51)

#Eğitim setiyle model kurulması

model = model.fit(X\_train, y\_train)

#Kurulan model'in test edilmesi

y\_pred = model.predict(X\_test)

#Çıkan doğruluk skoru ve Hata Matrisi

cm = confusion\_matrix(y\_test, y\_pred)

print("confusion\_matrix:\n", cm)

print("accuracy\_score: ", accuracy\_score(y\_test, y\_pred))

Burada aldığımız sonuç ise :

confusion\_matrix:

[[909 78]

[141 164]]

accuracy\_score: 0.8304953560371517

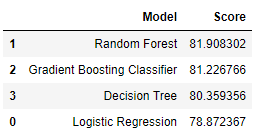
Ardından bu model'i sürekli birdaha eğitmemek için eğitilmiş model'i pickle aracılığıyla çıktı olarak aldım.

filename = 'traditionalml.sav'

pickle.dump(model, open(filename, 'wb'))

**Risk Tabanlı Makine Öğrenmesi Algoritması**

Bu algoritma'da da ön işlemelere kadar herşey aynı, burada ise farklı algoritmalar test edip sonuçlarına baktım. Sonuçlar şu şekildeydi :



Buradaki fark, kullanıcılar için direkt 0-1 değeri almak yerine 0 ile 1 arasında bir risk skoru alıyor, ardından bunu bir eşik değeri ile ben 0 ve 1'e dönüştürüyordum. Buradaki amaç ise şirket'in kar maksimizasyonunu sağlayacak olan yaklaşımı kurgulayabilmek.

#Burada yazdığım fonksiyon yeni girdiyi makineye uygun hale getiriyor.

def preProcess(a):

    data=list(a.values())

    colz=list(a.keys())

    dfx=pd.DataFrame(data=[data], columns=colz)

    XX1 =np.array(dfx)

    XX2=dfx[features]

    XX = np.hstack((XX1))

    return XX

#Bu fonksiyon ise bahsettiğim eşik değerini belirleyip kullanıcıları 0 veya 1 'e atıyor.

def cutoff(x):

    if x > 0.50:

        return 1

    if x <= 0.50:

        return 0

Şimdiyse bu algoritmayı X\_test yani test verisetimiz de deneyelim.

test = X\_test

#Bu kısımda tek tek bütün satırlardaki girdi değerlerini okuyarak, onlar için bir risk skoru atıyor.

X\_testindex = list(X\_test.index)

probs = []

for i in X\_testindex:

    a = {'CEK\_TUTAR':test.CEK\_TUTAR[i], 'VADE\_GUN':test.VADE\_GUN[i], 'SON5ISTIHBARAT\_SONUC':test.SON5ISTIHBARAT\_SONUC[i],

         'SON5POLITIKA\_SONUC':test.SON5POLITIKA\_SONUC[i],'SON5CEK\_RENK\_ORTALAMASI':test.SON5CEK\_RENK\_ORTALAMASI[i],

         'SON5KULLANDIRIM':test.SON5KULLANDIRIM[i], 'ŞUBE':test.ŞUBE[i], 'SIRKET\_TURU':test.SIRKET\_TURU[i],

         'MUSTERI\_RISK\_SEVIYESI':test.MUSTERI\_RISK\_SEVIYESI[i], 'TK\_KURUMSAYISISON5':test.TK\_KURUMSAYISISON5[i],

         'TK\_GECIKMEHESAPSON5':test.TK\_GECIKMEHESAPSON5[i],

         'TK\_GECIKMEBAKIYEORT':test.TK\_GECIKMEBAKIYEORT[i], 'TK\_NAKDILIMIT':test.TK\_NAKDILIMIT[i],

         'TK\_NAKDIRISK':test.TK\_NAKDIRISK[i],'TK\_GAYRINAKDILIMIT':test.TK\_GAYRINAKDILIMIT[i],

         'BK\_KURUMSAYISI':test.BK\_KURUMSAYISI[i],'BK\_LIMIT':test.BK\_LIMIT[i],

         'BK\_RISK':test.BK\_RISK[i],'BK\_GECIKMEHESAP':test.BK\_GECIKMEHESAP[i],

         'BK\_GECIKMEBAKIYE':test.BK\_GECIKMEBAKIYE[i],'T2\_NAKDILIMIT\_TPL':test.T2\_NAKDILIMIT\_TPL[i],

         'T2\_NAKDIRISK\_TPL':test.T2\_NAKDIRISK\_TPL[i], 'T2\_NAKDIRISK\_KV':test.T2\_NAKDIRISK\_KV[i],

         'T2\_GAYRINAKDILIMIT':test.T2\_GAYRINAKDILIMIT[i],'T2\_GAYRINAKDIRISK':test.T2\_GAYRINAKDIRISK[i],

         'T2\_FKTRNG\_LIMIT':test.T2\_FKTRNG\_LIMIT[i],'T2\_FKTRNG\_TPLRISK':test.T2\_FKTRNG\_TPLRISK[i],

         'T2\_NAKDIRISK\_OV':test.T2\_NAKDIRISK\_OV[i],'T2\_NAKDIRISK\_UV':test.T2\_NAKDIRISK\_UV[i],

         'T2\_SORUNLUKOD\_RISK':test.T2\_SORUNLUKOD\_RISK[i],'T3\_NAKDILIMIT\_TPL':test.T3\_NAKDILIMIT\_TPL[i],

         'T3\_NAKDIRISK\_TPL':test.T3\_NAKDIRISK\_TPL[i],'T3\_NAKDIRISK\_KV':test.T3\_NAKDIRISK\_KV[i],

         'T3\_NAKDIRISK\_OV':test.T3\_NAKDIRISK\_OV[i], 'T3\_NAKDIRISK\_UV':test.T3\_NAKDIRISK\_UV[i],

         'T3\_GAYRINAKDILIMIT':test.T3\_GAYRINAKDILIMIT[i],'T3\_GAYRINAKDIRISK':test.T3\_GAYRINAKDIRISK[i],

         'T3\_FKTRNG\_LIMIT':test.T3\_FKTRNG\_LIMIT[i],'T3\_SORUNLUKOD\_RISK':test.T3\_SORUNLUKOD\_RISK[i],

         'T3\_FKTRNG\_TPLRISK':test.T3\_FKTRNG\_TPLRISK[i], 'T4\_NAKDILIMIT\_TPL':test.T4\_NAKDILIMIT\_TPL[i],

         'T4\_NAKDIRISK\_TPL':test.T4\_NAKDIRISK\_TPL[i],'T4\_NAKDIRISK\_KV':test.T4\_NAKDIRISK\_KV[i],

         'T4\_NAKDIRISK\_OV':test.T4\_NAKDIRISK\_OV[i],'T4\_NAKDIRISK\_UV':test.T4\_NAKDIRISK\_UV[i],

         'T4\_GAYRINAKDILIMIT':test.T4\_GAYRINAKDILIMIT[i],'T4\_GAYRINAKDIRISK':test.T4\_GAYRINAKDIRISK[i],

         'T4\_FKTRNG\_LIMIT':test.T4\_FKTRNG\_LIMIT[i],'T4\_SORUNLUKOD\_RISK':test.T4\_SORUNLUKOD\_RISK[i],

         'T4\_FKTRNG\_TPLRISK':test.T4\_FKTRNG\_TPLRISK[i]}

#Girdi için uygun hale getirilme işlemi

XX=preProcess(a)

    XX = XX.reshape(1,-1)

#Şimdi ise girdi için risk skoru atanacak.

    probs.append(grd.predict\_proba(XX)[:,1][0])

#Bütün sonuçlar Atandıktan sonra bunlar bir dataframe'ye çekiliyor. Ve ardından Eşik değeri ile yeniden düzenleniyor.

X\_testindex = pd.DataFrame(X\_testindex)

probs = pd.DataFrame(probs)

testler = pd.concat([probs,X\_testindex],axis=1)

testler.columns = ["Probs","İndex"]

testler = testler.set\_index("İndex")

#Buradaki WFunc Cut-Off ile kesilmiş ve yeniden etiketlenmiş yeni değerler

testler["WFunc"] = None

testler["WFunc"] = testler.Probs.apply(cutoff)

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, testler.WFunc)

print("confusion\_matrix:\n", cm)

print("accuracy\_score: ", accuracy\_score(y\_test, testler.WFunc))

Bu yaklaşımda biz Eşik değerini 0.5 Kullandık, bu değer problemden probleme değişebiliyor, Aynı zamanda değişimde Bir yandan Yalnış Etiketledikleriniz ile Yakalama oranınız arasındaki trade-off'a etki ediyor.

confusion\_matrix:

[[1128 83]

[ 220 183]]

accuracy\_score: 0.8122676579925651

Ardından bu modelin de daha sonra birdaha eğitmeden kullanabilmek için pickle ile çıktısını aldım.

filename = 'olasilikbazlimodel.sav'

pickle.dump(grd, open(filename, 'wb'))

**Keras Bazlı Derin Öğrenme Yaklaşımı**

Bu algoritma'da da ön işlemelere kadar herşey aynı, burada yolladığımız veriyi (-1,1) arasında ölçeklendirip yollamamız gerekiyor. O işlemide şu şekilde yapıyoruz :

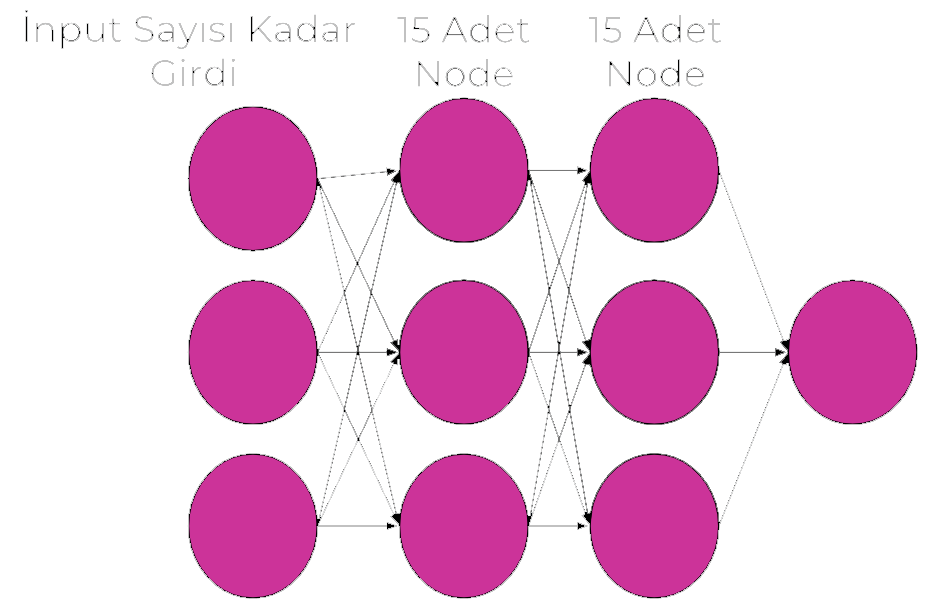
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler((-1,1))

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

Ardından derin öğrenme modelimizi kuruyoruz. Burada kurduğumuz model şuna benzeyecek :



Kodu ise şu şekilde :

#Burada Batch(Parça Sayısı) değişkeni için 200 , Epoch(Eğitim Turu Sayısı) değişkeni için 1800 kullanıldı, yani model 1800 kere verisetine bakıyor.

batch\_size = 200

nb\_epoch = 1800

model = Sequential()

#Burası giriş katmanı

model.add(Dense(55, input\_shape=(50,)))

model.add(Activation("hard\_sigmoid"))

model.add(BatchNormalization())

model.add(Dropout(0.1))

#Buradakiler ise ara katmanlar 15 adet node bulunduruyorlar.)

model.add(Dense(15))

model.add(Activation("hard\_sigmoid"))

model.add(BatchNormalization())

model.add(Dropout(0.1))

model.add(Dense(15))

model.add(Activation("hard\_sigmoid"))

model.add(BatchNormalization())

model.add(Dropout(0.1))

#Çıkış katmanı 0-1 arasında olasılık değeri döndürüyor.

model.add(Dense(1))

model.add(Activation('sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam')

#Modeli fit ediyoruz.

model.fit(X\_train\_scaled, y\_train, batch\_size=batch\_size, epochs=nb\_epoch, verbose=0)

Ardından eğittimiz modele test verisetini yollayarak tahmin değerlerini aldık, aldığımız tahmin değerleri 0-1 arasında olduğundan dolayı bunu 0.5'ten yukarısı 1 aşağısı 0 şeklinde kestik. Ve Tahminlerimizi gerçek sonuçlar ile karşılaştırdık.

y\_pred = model.predict(X\_test\_scaled)

y\_pred = (y\_pred > 0.5) \* 1

cm = confusion\_matrix(y\_test, y\_pred)

print("confusion\_matrix:\n", cm)

print("accuracy\_score: ", accuracy\_score(y\_test, y\_pred))

confusion\_matrix:

[[715 272]

[ 98 207]]

accuracy\_score: 0.7136222910216719

Burada ise yine aynı şekilde birdaha eğitmemek için kaydettik. Ama burada pickle kullanmadık ve modeli 2 farklı dosyaya böldük.

* Model'in Ağırlıkları
* Model'in Mimarisi

model.save\_weights('model\_weights.h5')

with open('model\_architecture.json', 'w') as f:

    f.write(model.to\_json())

Şimdi ise topluluk oylamasını nasıl kodladığımıza geçelim.

**Topluluk Oylaması / Majority Vote Algoritması**

Buradaki çalışma mantığını daha önce anlatmıştım, kodlardan bahsedeyim.

1. Test Verisetini Import edip belli ön işlemeler yapacağız(Kategorik veriler için)
2. Kaydettiğimiz bütün modelleri çağırarak test edeceğiz.
3. Topluluk oylaması algoritmasını kurgulayacağız. Ve test edip sonuçlarına bakacağız.

Evet başlayalım.

Öncelikle gerekli kütüphaneler ve Test verisetini import edelim.

import pickle

import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn.metrics import confusion\_matrix, accuracy\_score

from keras.models import model\_from\_json

from sklearn.preprocessing import MinMaxScaler

#Verisetinin import edilmesi, duplicate değerler'in düşürülmesi

test = pd.read\_excel("X\_test.xlsx")

test = test.drop\_duplicates()

target = test.Churn

test.drop("Churn", axis=1, inplace=True)

features = list(test.columns)

#Kategorik verilerin numerik hale getirilmesi

le = preprocessing.LabelEncoder()

objects = ["SON5ISTIHBARAT\_SONUC","SON5POLITIKA\_SONUC","SON5CEK\_RENK\_ORTALAMASI","SON5KULLANDIRIM","ŞUBE","SIRKET\_TURU"]

for i in objects:

    test[i] = le.fit\_transform(test[i])

Ardından kaydettiğimiz modelleri çağırarak test edelim.

**Geleneksel ML**

traditionalml = pickle.load(open('traditionalml.sav', 'rb'))

traditionalml

y\_pred = traditionalml.predict(test)

cm = confusion\_matrix(target, y\_pred)

print("confusion\_matrix:\n", cm)

print("accuracy\_score: ", accuracy\_score(target, y\_pred))

confusion\_matrix:

[[1110 62]

[ 182 221]]

accuracy\_score: 0.8450793650793651

**Olasılık Bazlı ML**

olasilikbazli = pickle.load(open('olasilikbazlimodel.sav', 'rb'))

olasilikbazli

def preProcess(a):

    data=list(a.values())

    colz=list(a.keys())

    dfx=pd.DataFrame(data=[data], columns=colz)

    XX1 =np.array(dfx)

    XX2=dfx[features]

    XX = np.hstack((XX1))

    return XX

def cutoff(x):

    if x >= 0.50:

        return 1

    if x <= 0.50:

        return 0

X\_testindex = list(test.index)

probs = []

for i in X\_testindex:

    a = {'CEK\_TUTAR':test.CEK\_TUTAR[i], 'VADE\_GUN':test.VADE\_GUN[i], 'SON5ISTIHBARAT\_SONUC':test.SON5ISTIHBARAT\_SONUC[i],

         'SON5POLITIKA\_SONUC':test.SON5POLITIKA\_SONUC[i],'SON5CEK\_RENK\_ORTALAMASI':test.SON5CEK\_RENK\_ORTALAMASI[i],

         'SON5KULLANDIRIM':test.SON5KULLANDIRIM[i], 'ŞUBE':test.ŞUBE[i], 'SIRKET\_TURU':test.SIRKET\_TURU[i],

         'MUSTERI\_RISK\_SEVIYESI':test.MUSTERI\_RISK\_SEVIYESI[i], 'TK\_KURUMSAYISISON5':test.TK\_KURUMSAYISISON5[i],

         'TK\_GECIKMEHESAPSON5':test.TK\_GECIKMEHESAPSON5[i],

         'TK\_GECIKMEBAKIYEORT':test.TK\_GECIKMEBAKIYEORT[i], 'TK\_NAKDILIMIT':test.TK\_NAKDILIMIT[i],

         'TK\_NAKDIRISK':test.TK\_NAKDIRISK[i],'TK\_GAYRINAKDILIMIT':test.TK\_GAYRINAKDILIMIT[i],

         'BK\_KURUMSAYISI':test.BK\_KURUMSAYISI[i],'BK\_LIMIT':test.BK\_LIMIT[i],

         'BK\_RISK':test.BK\_RISK[i],'BK\_GECIKMEHESAP':test.BK\_GECIKMEHESAP[i],

         'BK\_GECIKMEBAKIYE':test.BK\_GECIKMEBAKIYE[i],'T2\_NAKDILIMIT\_TPL':test.T2\_NAKDILIMIT\_TPL[i],

         'T2\_NAKDIRISK\_TPL':test.T2\_NAKDIRISK\_TPL[i], 'T2\_NAKDIRISK\_KV':test.T2\_NAKDIRISK\_KV[i],

         'T2\_GAYRINAKDILIMIT':test.T2\_GAYRINAKDILIMIT[i],'T2\_GAYRINAKDIRISK':test.T2\_GAYRINAKDIRISK[i],

         'T2\_FKTRNG\_LIMIT':test.T2\_FKTRNG\_LIMIT[i],'T2\_FKTRNG\_TPLRISK':test.T2\_FKTRNG\_TPLRISK[i],

         'T2\_NAKDIRISK\_OV':test.T2\_NAKDIRISK\_OV[i],'T2\_NAKDIRISK\_UV':test.T2\_NAKDIRISK\_UV[i],

         'T2\_SORUNLUKOD\_RISK':test.T2\_SORUNLUKOD\_RISK[i],'T3\_NAKDILIMIT\_TPL':test.T3\_NAKDILIMIT\_TPL[i],

         'T3\_NAKDIRISK\_TPL':test.T3\_NAKDIRISK\_TPL[i],'T3\_NAKDIRISK\_KV':test.T3\_NAKDIRISK\_KV[i],

         'T3\_NAKDIRISK\_OV':test.T3\_NAKDIRISK\_OV[i], 'T3\_NAKDIRISK\_UV':test.T3\_NAKDIRISK\_UV[i],

         'T3\_GAYRINAKDILIMIT':test.T3\_GAYRINAKDILIMIT[i],'T3\_GAYRINAKDIRISK':test.T3\_GAYRINAKDIRISK[i],

         'T3\_FKTRNG\_LIMIT':test.T3\_FKTRNG\_LIMIT[i],'T3\_SORUNLUKOD\_RISK':test.T3\_SORUNLUKOD\_RISK[i],

         'T3\_FKTRNG\_TPLRISK':test.T3\_FKTRNG\_TPLRISK[i], 'T4\_NAKDILIMIT\_TPL':test.T4\_NAKDILIMIT\_TPL[i],

         'T4\_NAKDIRISK\_TPL':test.T4\_NAKDIRISK\_TPL[i],'T4\_NAKDIRISK\_KV':test.T4\_NAKDIRISK\_KV[i],

         'T4\_NAKDIRISK\_OV':test.T4\_NAKDIRISK\_OV[i],'T4\_NAKDIRISK\_UV':test.T4\_NAKDIRISK\_UV[i],

         'T4\_GAYRINAKDILIMIT':test.T4\_GAYRINAKDILIMIT[i],'T4\_GAYRINAKDIRISK':test.T4\_GAYRINAKDIRISK[i],

         'T4\_FKTRNG\_LIMIT':test.T4\_FKTRNG\_LIMIT[i],'T4\_SORUNLUKOD\_RISK':test.T4\_SORUNLUKOD\_RISK[i],

         'T4\_FKTRNG\_TPLRISK':test.T4\_FKTRNG\_TPLRISK[i]}

    XX=preProcess(a)

    XX = XX.reshape(1,-1)

    probs.append(olasilikbazli.predict\_proba(XX)[:,1][0])

X\_testindex = pd.DataFrame(X\_testindex)

probs = pd.DataFrame(probs)

testler = pd.concat([probs,X\_testindex],axis=1)

testler.columns = ["Probs","İndex"]

testler = testler.set\_index("İndex")

testler["WFunc"] = None

testler["WFunc"] = testler.Probs.apply(cutoff)

cm = confusion\_matrix(target, testler.WFunc)

print("confusion\_matrix:\n", cm)

print("accuracy\_score: ", accuracy\_score(target, testler.WFunc))

confusion\_matrix:

[[1093 79]

[ 220 183]]

accuracy\_score: 0.8101587301587302

**Derin Öğrenme Yaklaşımı**

with open('model\_architecture.json', 'r') as f:

    kerasmodel = model\_from\_json(f.read())

kerasmodel.load\_weights('model\_weights.h5')

def KerasPreProcess(test):

    scaler = MinMaxScaler((-1,1))

    return scaler.fit\_transform(test)

def scalers(x):

    if x > 0.5:

        return 1

    if x <= 0.5:

        return 0

test2 = KerasPreProcess(test)

y\_pred = kerasmodel.predict(test2)

y\_pred = (y\_pred > 0.5) \* 1

cm = confusion\_matrix(target, y\_pred)

print("confusion\_matrix:\n", cm)

print("accuracy\_score: ", accuracy\_score(target, y\_pred))

confusion\_matrix:

[[1078 94]

[ 219 184]]

accuracy\_score: 0.8012698412698412

Bu kısımda çoğu şeyi fonksiyonlar üzerinden kurguladım sebebi ise hepsini majority vote algoritmasının içinde kullanacağım. Ve daha temiz bir kod olsun istedim.

**Majority Vote Algoritması**

def MajorityVote(test):#OLasılık tabanlı için tahminler yapılıyor.

olasiliksonuc = listdegerverme(list(test.index))

#geleneksel Ml için tahminler yapılıyor.

    gelenekselml = traditionalml.predict(test)

#keras için tahminler yapılıyor.

    kerassonuc = kerasmodel.predict(KerasPreProcess(test))

    for i in range(0,len(kerassonuc)):

        kerassonuc[i] = scalers(kerassonuc[i])

#Ardından bu 3 sonucuda dataframe haline getirip yan yana yazdırıyoruz.

    kerassonuc = pd.DataFrame(kerassonuc)

    gelenekselml = pd.DataFrame(gelenekselml)

    index = pd.DataFrame(list(test.index))

    gelenekselml = pd.concat([gelenekselml,index], axis=1)

    gelenekselml.columns = ["Sonuc","İndex"]

    gelenekselml.set\_index("İndex")

    olasiliksonuc = pd.DataFrame(olasiliksonuc)

    gelenekselml = gelenekselml.Sonuc

    majority = pd.concat([olasiliksonuc,gelenekselml,kerassonuc],axis=1)

#Burada ise son olarak Majority diye bir kolon ekliyoruz.

    majority["Majority"] = None

    majority.columns = ["Olasilik","Geleneksel","Keras","Majority"]

#Majority kolonu tek tek şunlara bakılarak dolduruluyor.#Eğer 3 Model 1 dediyse toplam 3 oluyor ve kişi Churn oluyor#Eğer 2 Model 1 dediyse toplam 2 oluyor ve kişi Churn oluyor.#Eğer 1 Model 1 dediyse toplam 1 oluyor ve kişi Churn olmuyor.

    for i in range(0,len(majority)):

        if majority.Olasilik[i] + majority.Geleneksel[i] + majority.Keras[i] == 3:

            majority.Majority[i] = 1

        elif majority.Olasilik[i] + majority.Geleneksel[i] + majority.Keras[i] == 2:

             majority.Majority[i] = 1

        elif majority.Olasilik[i] + majority.Geleneksel[i] + majority.Keras[i]== 1:

             majority.Majority[i] = 0

        elif majority.Olasilik[i] + majority.Geleneksel[i] + majority.Keras[i]== 0:

            majority.Majority[i] = 0

    return majority.Majority

Ve algoritmayı çalıştırıp sonuçları alalım.

Majority = MajorityVote(test)

Majority = pd.DataFrame(Majority)

index = pd.DataFrame(list(test.index))

Majority = pd.concat([Majority,index],axis=1)

Majority.columns = ["Majority","İndex"]

Majority = Majority.set\_index("İndex")

Majority = Majority.astype(int)

cm = confusion\_matrix(target, Majority)

print("confusion\_matrix:\n", cm)

print("accuracy\_score: ", accuracy\_score(target, Majority))

Sonuçlar ise şu şekilde :

confusion\_matrix:

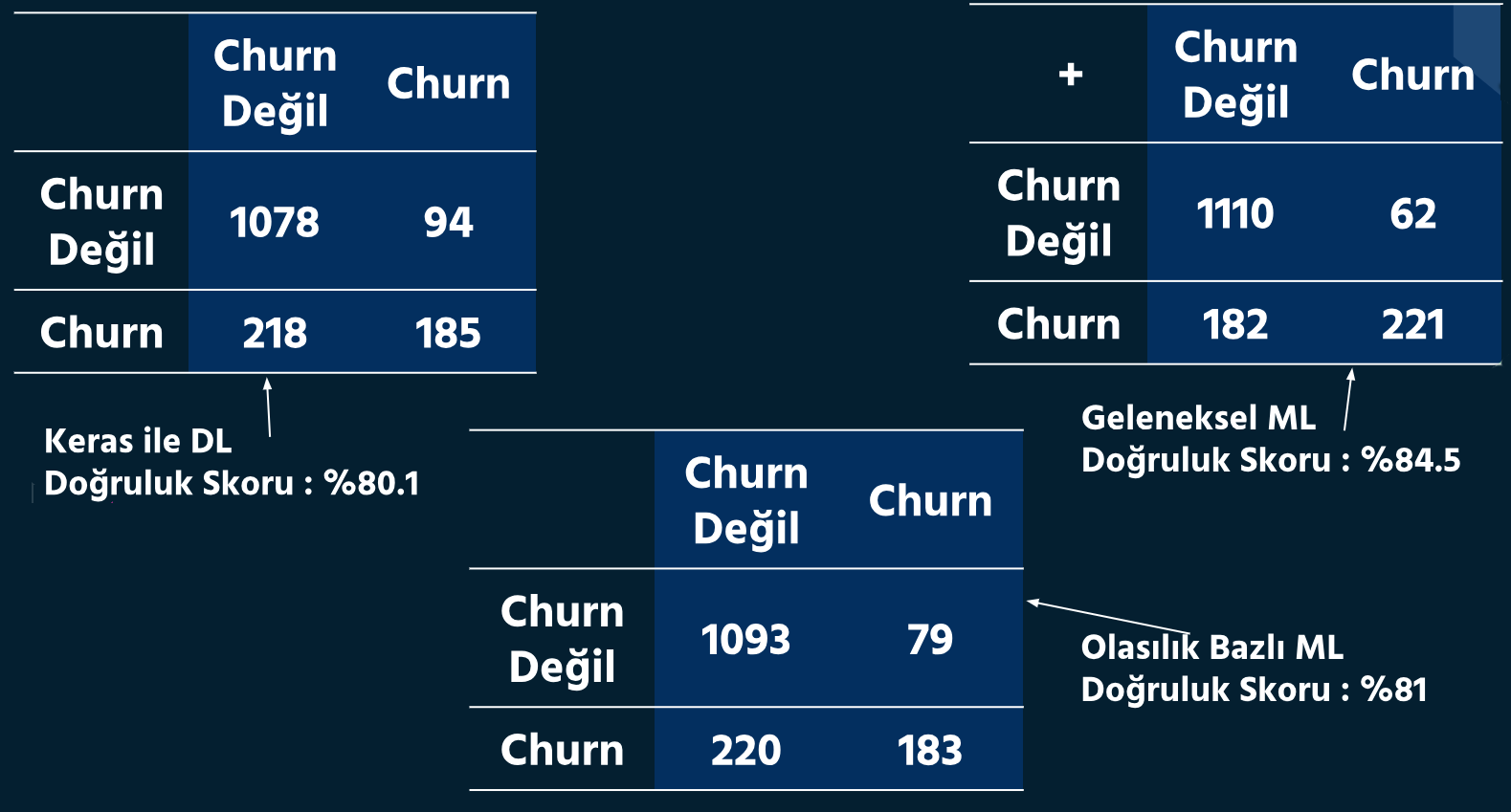
[[1098 74]

[ 193 210]]

accuracy\_score: 0.8304761904761905

**Genel Sonuçlar**

Bütün Modellerin Sonuçlarına Bakmak Gerekirse :



Ve son Olarak Majority Vote Skorumuz ise :

