**CHURN ANALİZİ İLE MÜŞTERİ KAYBININ TESPİTİ**

**İlim Betül Yavuz**

**ÖZET**

Günümüzde çoğu kurumsal firmanın rekabet ortamında mevcut müşteriyi mutlu ederek elde tutmak yeni müşteri kazanmaktan daha önemlidir. Müşteri hareketlerini inceleyerek terk etme ihtimali olan müşterilere ait profilleri çıkarıp terk etme olasılığı fazla olan müşterileri daha önceden tahmin etme işlemi Churn Analizi (Müşteri Kaybı Analizi) olarak adlandırılmaktadır. Churn analizinin amacı, müşterinin firmayı terk etmeden ya da terk etmeye teşebbüs etmeden önce bu durumun farkında olmak ve ardından bu durumu önleyici çalışmalar halinde bulunmaktır. Var olan bir problemi istenilen amaç dahilinde çeşitli çözme yöntemleri vardır. Churn analizinde tahmin(regression) ya da sınıflandırma(classification) yöntemleri kullanılabilir. Projede sınıflandırma yöntemleri kullanılmıştır. Müşterinin terk etme durumu 1, terk etmeme durumu 0 olarak sınıflandırılacaktır. Veriseti Kaggle’dan temin edilmiştir. 10000 adet veriden oluşmaktadır

**VERİSETİ İNCELEMESİ**

**Veri Link:** <https://www.kaggle.com/shrutimechlearn/churn-modelling>

**Age**:Yaş

**Tenure**: Ne kadar süredir firmada olduğu

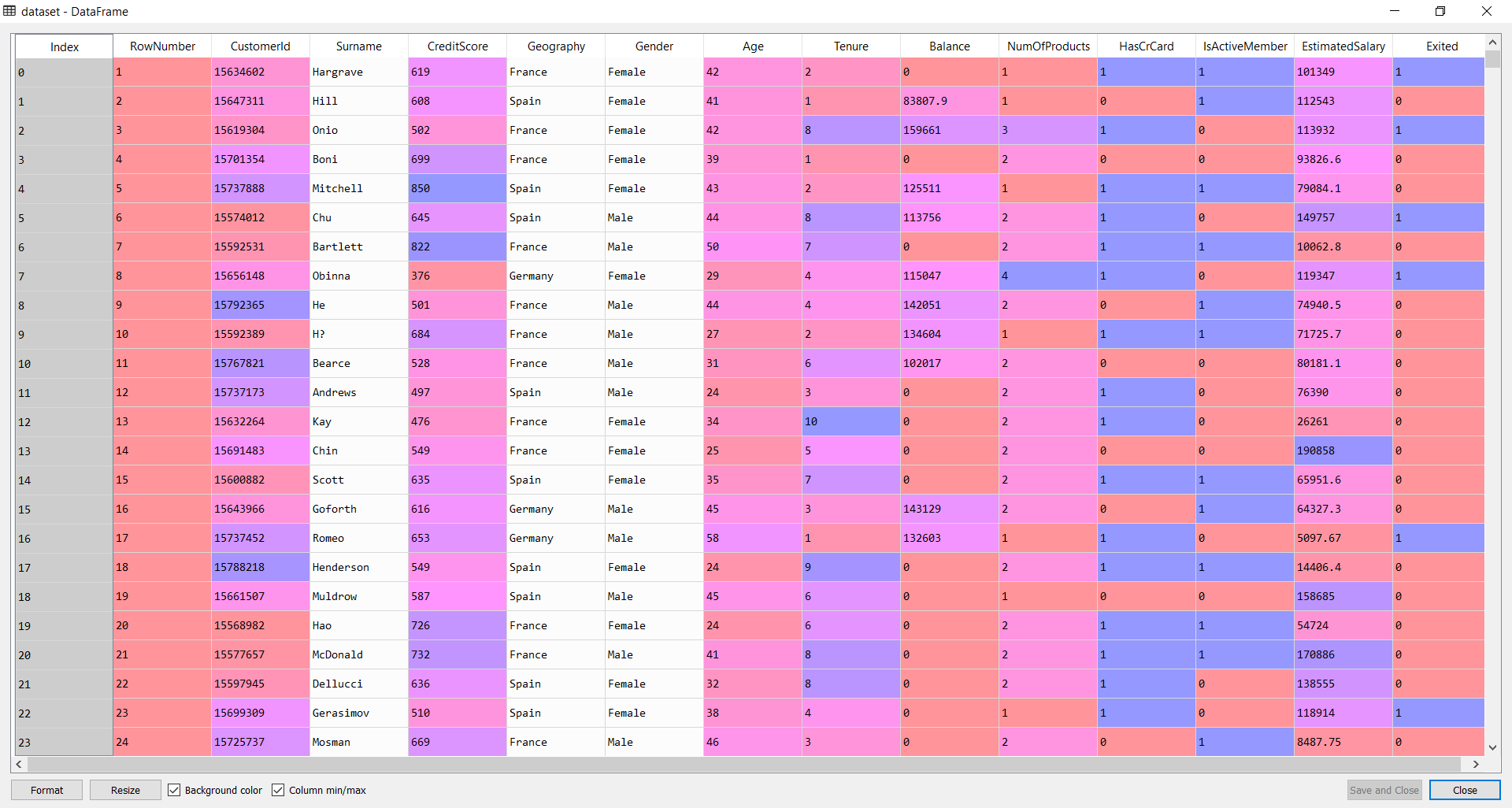
**Balance**: Ne kadarlık hesabı olduğu

**NumOfProducts**: Kaç ürün aldığı

**HasCr**: Kredi kartı olup olmadığı

**IsActive**: Aktif olup olmadığı

**EstimatedSalary**: Tahmini maaş

**Exited**: Bırakıp bırakmadığı 

Şekil 1: Veri Seti

**CHURN ANALİZİ (KODLAR)**

**import pandas as pd**

**import numpy as np**

**dataset = pd.read\_csv('Churn\_Modelling.csv')**

>>Pandas kütüphanesi ile dizinde bulunan veri seti projeye import edilir.

**from sklearn.preprocessing import LabelEncoder**

**for column in dataset.columns:**

**if dataset[column].dtype == np.number:**

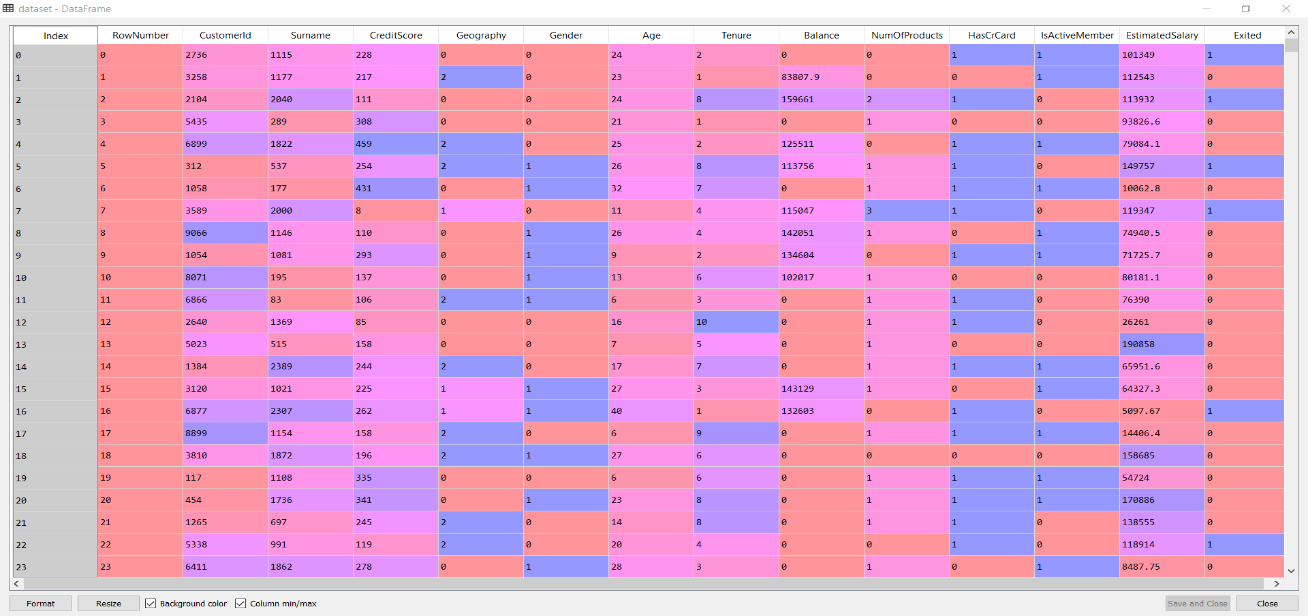
**continue**

**dataset[column] = LabelEncoder().fit\_transform(dataset[column])**

>>Sklearn kütüphanesindeki ön işleme kısmından LabelEncoder ile sayısal olmayan kategorik veriler sayısal hale dönüştürülür.

**X = dataset.iloc[:, 3:13].values** #3.kolondan itibaren son kolon hariç alınır

**y = dataset.iloc[:, 13].values**  #son kolon(hesaplanması istenen kolon)

>> Amaç, bağımsız değişken olan X değerlerinden bağımlı değişken olan Y değişkenini bulmak ve elimizdeki Y değişkenleri ile karşılaştırıp ne kadar doğru sonuç elde edildiğini görmektir. Yapay sinir ağının ezber yapması (overfitting) istenmediği için RowNumber, Custumerid, Surname gibi unique olabilecek kolonlar veri setinden çıkarılır. 

Şekil 2:Düzenlemeler Sonrası Veri Seti

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

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.33, random\_state = 0)**

>>Model eğitim ve test veri seti olarak bölünür. Test veri seti ‘test\_size’ ile eldeki verinin 0.33 ‘ü olacak şekilde ayarlanır, random\_state = 0 ile tekrar eden durumlarda aynı değer verilmesi sağlanır.

**from sklearn.preprocessing import StandardScaler**

**sc\_X = StandardScaler()**

**X\_train = sc\_X.fit\_transform(X\_train)**

**X\_test = sc\_X.transform(X\_test)**

>> Makine 0-1 arasındaki değerlere işlem yapabilir. Bu yüzden Scaling(ölçeklendirme) yapılır. Probleme göre çeşitli ölçeklendirme yöntemleri kullanılabilir. Bunlardan bazıları Standardizasyon, Normalizasyon, Min-Max ölçekleme vb yöntemlerdir. Projede Standardizayon yöntemi kullanılmıştır.

**from keras.callbacks import History,EarlyStopping**

**mycallbacks = EarlyStopping(monitor='loss', patience=2,verbose=1)**

>>CallBack(geri çağrı) öğrenme tamamlandığında bize projenin bilgi vermesini, içsel durumunu görülmesini sağlar. EarlyStopping (Erken Durdurma) yöntemi kullanılarak eğer öğrenme artık gerçekleşmiyorsa ve ezberleme durumuna geçilmeye başlandıysa bu yöntem ile eğitim sonlanır. İsteğe göre ayrıntılı bir şekilde sonuç gösterilir ve ağırlıklar yeniden ayarlanır.

**from keras.models import Sequential**

**from keras.layers import Dense**

**from keras.layers.core import Activation**

**from keras.layers.core import Dropout**

**from keras import losses**

**model=Sequential()** # Sequential model oluşturmak için

**model.add(Dense(256,activation="relu", input\_dim=10, kernel\_initializer="uniform"))** # giriş katmanı

**model.add(Dense(128,activation="relu", kernel\_initializer="uniform"))** #gizli katman

**model.add(Dense(1,activation="sigmoid", kernel\_initializer="uniform"))** #çıkış katmanı

>>3 katmandan oluşan bir yapay sinir ağı oluşturulur; giriş (1), gizli (1), çıkış (1). Giriş katmanında 256 hücre, aktivasyon fonksiyonu olarak ‘relu ’ vardır. Ne kadar kolon işleme sokulacaksa ‘input\_dim’ ile belirtilir. Gizli katmanda 128 hücre, aktivasyon fonksiyonu olarak ‘relu ’vardır. Çıkış katmanında 1 hücre (1 çıkış alınacağı için), aktivasyon fonksiyonu olarak ‘sigmoid’ vardır. Sinir Ağının Modeli probleme ya da tasarlayan kişinin tercihine göre değişmektedir.

 ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 3 : ReLU Şekil 4 : Sigmoid

**model.compile(optimizer='adam', loss='binary\_crossentropy', metrics = ['accuracy'])**

**history=model.fit(X\_train, y\_train,validation\_split=0.2, batch\_size=X\_train.shape[0], epochs=200, callbacks=[mycallbacks], verbose=1)**

>>Modeli hazırlarken öğrenme işlemi için gerekli konfigürasyonlar yapılır. Bunlar:

Optimizer: Ağırlık katsayılarının Güncellenmesi için kullanılıcak optimiasyon yöntemleridir. Bunlardan bazıları: SGD,Adam ,Nadam, Adamax, AdaGrad vs..

Loss function: Gerçek değer ile tahmin edilen değer arasındaki hatayı ifade eder.

Bunlardan bazıları: Binary-Cross-entropy, Categorical-cross-entropy,mean-squared-entropy vb..

Metrics: Eğitim ve test arasındaki değerlendirme parametreleridir.

Bunlar; mae,msc,acc vs..

>>validation\_split ile eğitim setinin 0.2 sinin validation olarak verilmiştir. Validation değerler ile aslında eğitim verisinin bir ön değerlendirmesi yapılır. Epoch değeri ile eğitimin kaç çevrim olacağı belirtilir. Callback değerine belirlediğimiz yapı eşitlenir. Verbose=1 ile ayrıntılı görünüm istenir.

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

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

>>Sonuçların tahmini için predict fonksiyonu kullanılır, test verisi parametresi olur. Eğer tahmin değeri 0.5 ‘den büyükse sonuç olarak alınır.

**from sklearn.metrics import confusion\_matrix**

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

**print(cm)**

>>Confusion Matrics oluşturulur. Böylelikle ne kadar veriden ne kadarının doğru ya da yanlış sonuçlandığı matrixde görünmüş olunur.

**import matplotlib.pyplot as plt**

**plt.plot(history.history['acc'])**

**plt.plot(history.history['val\_acc'])**

**plt.title('Model accuracy')**

**plt.ylabel('Accuracy')**

**plt.xlabel('Epoch')**

**plt.legend(['Train', 'Test'], loc='upper left')**

**plt.show()**

**plt.savefig("acc.png")**

>> History değişkenine eşitlenen modelin görselleştirilmesi için acc ve vall\_acc doğruluk değerlerinin grafiği çizilir ve kaydedilir.

**plt.plot(history.history['loss'])**

**plt.plot(history.history['val\_loss'])**

**plt.title('Model loss')**

**plt.ylabel('Loss')**

**plt.xlabel('Epoch')**

**plt.legend(['Train', 'Test'], loc='upper left')**

**plt.show(),**

**plt.savefig("loss.png")**

>>Modelin kayıp(loss) fonksiyonu görselleştirilmesi

**SONUÇ**

Train 5360 örnek, validate 1340 örnek;

**Sonuç 1:**

**epochs:500** seçildiğinde yapı ezberlemeye meyilli olmaya başladığı zaman erken durdurma (early stopping) gerçekleşmiştir. Early stopping gerçekleşen epoch 00376 tır.

**Epoch 00376: early stopping**

**Confusion Matrix:** 2450(sonucu 1 olup, 1 olarak sınıflandırılanlar) +329(sonucu 0 olup, 0 olarak sınıflandırılanlar) = 2.779‬ doğru sınıflandırılmıştır. 354+167=521 adet veri yanlış sınıflandırılmıştır.

**[[2450 167]**

**[ 354 329]]**

**ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, harita içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 5 :Accuracy Function Şekil 6:Loss Function

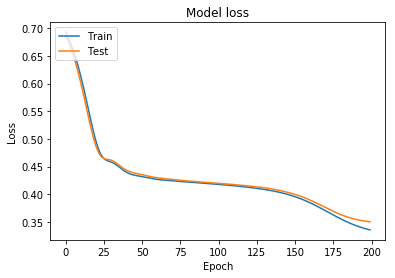
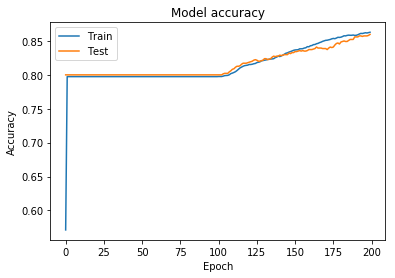
**Sonuç 2:**

**epochs:200** seçildiğinde yapıda ezberleme olmamaktadır, dolayısıyla herhangi bir Early Stopping durumu söz konusu değildir.

**Confusion Matrix:** 2507(sonucu 1 olup, 1 olarak sınıflandırılanlar) +328(sonucu 0 olup, 0 olarak sınıflandırılanlar) = 2835 doğru sınıflandırılmıştır. 355+110=465 adet veri yanlış sınıflandırılmıştır. Alınan sonuçlar incelendiğinde 1.Sonuca göre doğruluk payı daha yüksektir.

**[[2507 110]**

**[ 355 328]]**



Şekil 7:Loss Function Şekil 8:Accuracy Function