Modeling Earthquake Damage

Hüseyin PEKKAN  
*Artificial Intelligence Engineering*   
*TOBB ETU* Ankara,Türkiye   
201401016  
hpekkan@etu.edu.tr

*Abstract*—Bu dökümanda YAP470 dersi için hazırlanan Nepal’deki bir depremin verileri ve sonuçları ışığında yapılan bir makine öğrenmesi projesinin detayları görsellerle desteklenerek anlatılacak.

Keywords—veri bilimi, makine öğrenmesi, derin öğrenme,

# GİRİŞ

Makine öğrenmesi bilgisayarın kendisine verilen veriler arasında bazı benzerlikler bularak sınıflandırmaya çalışmasıdır. Bu sınıflandırmalar için çeşitli modeler kullanılmaktadır. Makine öğrenmesi kısmında verilerin mühendis tarafından işlenmesi gereklidir. Bu projede bir tane de derin öğrenme modeli kullanılmıştır.

# PROJE TANIMI

## Problem Tanımı

Bu projenin amacı 2015’de Nepal’in Gorkha Bölgesinde meydana gelen depremden sonra Nepal hükümeti tarafından toplanan veriler ışığında, hangi binaların ne kadar hasar alacağını tahmin etmektir. Ayrıca bu en büyük afet sonrası toplanan veridir. Eğitim verisinde ankete katılan toplamda 260601 bina (satır) ve 39 öznitelik (sütun) vardır. Amacımız bu veriler ışığında binanın alacağı hasarı tahmin etmektir. Bu hasarı azcan çoka 0, 1, 2 olarak sınıflandırmamız istenmekte.

Eğitilen modeler çeşitli f1 skor, accuracy, recall ve precison skorları ile karşılaştırıldı.

Yarışmada 0.756 f1 skora ulaşanlar vardı benim hedefim de 0.65 üzeri bir f1 skor elde etmekti.

# **Veri Seti, Veri özellikleri, Öznitelikler**

## Veri Seti, Veri özellikleri, Öznitelikler

Özniteliklere bu bağlantıdan ulaşabilirsiniz:

<https://www.drivendata.org/competitions/57/nepal-earthquake/page/136/>

damage\_grade: hasar miktarı

* 1 az hasar
* 2 orta hasar
* 3 çok hasar

geo\_level\_1\_id, geo\_level\_2\_id, geo\_level\_3\_id (int): genelden spesifiğe bölgeler. Muhtemel deperler: level 1:0-30, level 2: 0-1427, level 3: 0-12567

count\_floors\_pre\_eq (int): depremden önceki binadaki kat sayısı

age (int): bina yaşı

area\_percentage (int): binanın planının alanının normalize edilmiş hali

height\_percentage (int): binanın planının yüksekliğinin normalize edilmiş hali

land\_surface\_condition (kategorik): Binanın yapıldığı arazinin yüzey durumu. Olası değerler: n, o, t

foundation\_type (kategorik): İnşaat sırasında kullanılan temel türü. Olası değerler: h, i, r, u, w

roof\_type (kategorik): İnşaat sırasında kullanılan çatı tipi. Olası değerler: n, q, x

ground\_floor\_type (kategorik): zemin kat tipi. Olası değerler: f, m, v, x, z

other\_floor\_type (kategorik): zemin katlardan daha yüksekte kullanılan yapı türleri (çatı hariç). Olası değerler: j, q, s, x

position (kategorik): binanın konumu. Olası değerler: j, o, s, t.

plan\_configuration (kategorik): bina planı yapılandırması. Olası değerler: a, c, d, f, m, n, o, q, s, u

has\_superstructure\_adobe\_mud (binary): üst yapının Adobe/Çamurdan yapılıp yapılmadığını gösterir

has\_superstructure\_mud\_mortar\_stone (binary): Üst yapının Çamur Harç - Taştan yapılıp yapılmadığını gösterir.

has\_superstructure\_stone\_flag (binary): üst yapının Taştan yapılıp yapılmadığını gösterir

has\_superstructure\_cement\_mortar\_stone (binary): üst yapının Çimento Harcı - Taştan yapılıp yapılmadığını gösterir

has\_superstructure\_mud\_mortar\_brick (binary): üst yapının Çamur Harç - Tuğladan yapılıp yapılmadığını gösterir

has\_superstructure\_cement\_mortar\_brick (binary): üst yapının Çimento Harcı - Tuğladan yapılıp yapılmadığını gösterir

has\_superstructure\_timber (binary): üst yapının Keresteden yapılıp yapılmadığını gösterir

has\_superstructure\_bamboo (binary): üst yapının Bambudan yapılıp yapılmadığını gösterir

has\_superstructure\_rc\_non\_engineered (binary): üst yapının işlenmemiş betonarme olup olmadığını gösterir

has\_superstructure\_rc\_engineered (binary): üst yapının mühendislik betonarmesinden yapılıp yapılmadığını gösterir

has\_superstructure\_other (binary): üst yapının başka bir malzemeden yapılıp yapılmadığını gösterir

legal\_ownership\_status (kategorik): binanın yapıldığı arazinin yasal mülkiyet durumu. Olası değerler: a, r, v, w.

count\_families (int): Binada yaşayan aile sayısı

has\_secondary\_use (binary): binanın herhangi bir ikincil amaç için kullanılıp kullanılmadığını gösterir

has\_secondary\_use\_agriculture (binary): yapının tarımsal amaçlı kullanılıp kullanılmadığını gösterir

has\_secondary\_use\_hotel (binary): binanın otel olarak kullanılıp kullanılmadığını gösterir

has\_secondary\_use\_rental (binary): binanın kiralama amaçlı kullanılıp kullanılmadığını gösterir

has\_secondary\_use\_institution (binary): binanın herhangi bir kurumun yeri olarak kullanılıp kullanılmadığını gösterir

has\_secondary\_use\_school (binary): binanın okul olarak kullanılıp kullanılmadığını gösterir

has\_secondary\_use\_industry (binary): binanın endüstriyel amaçlarla kullanılıp kullanılmadığını gösterir

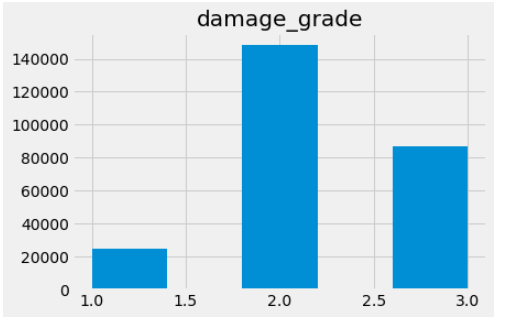
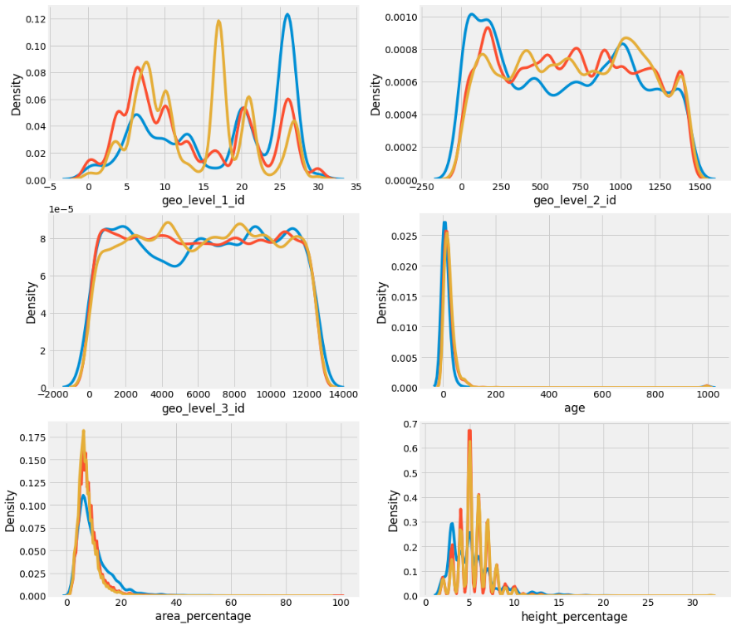
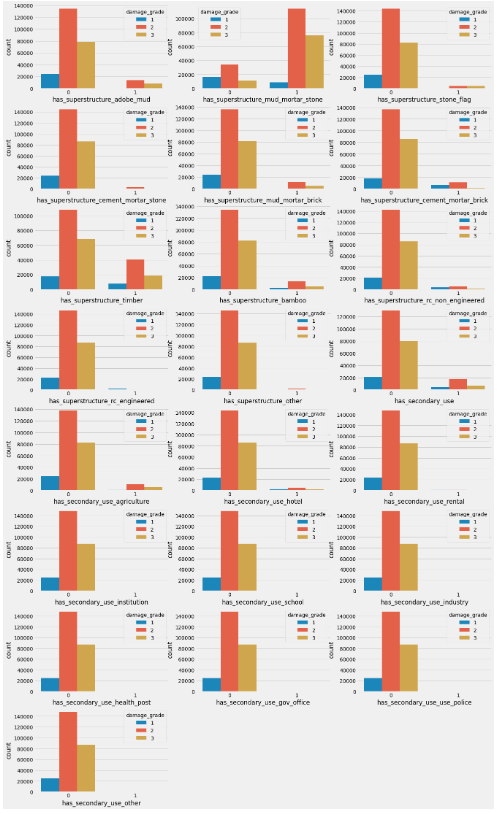
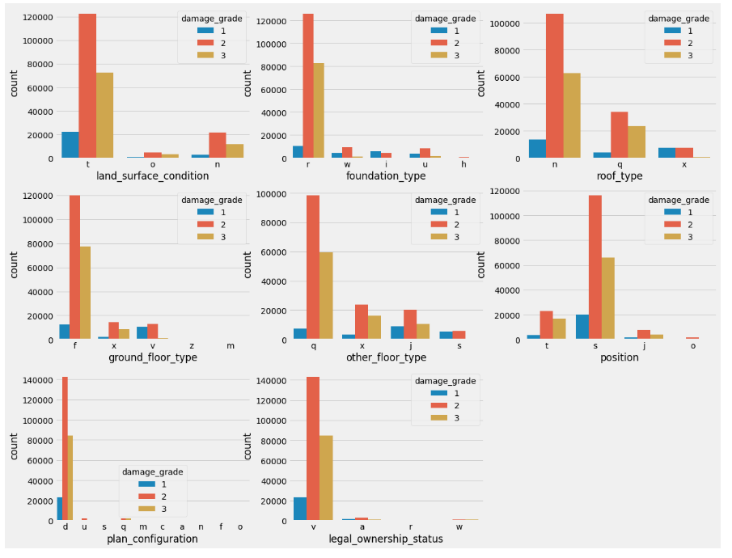
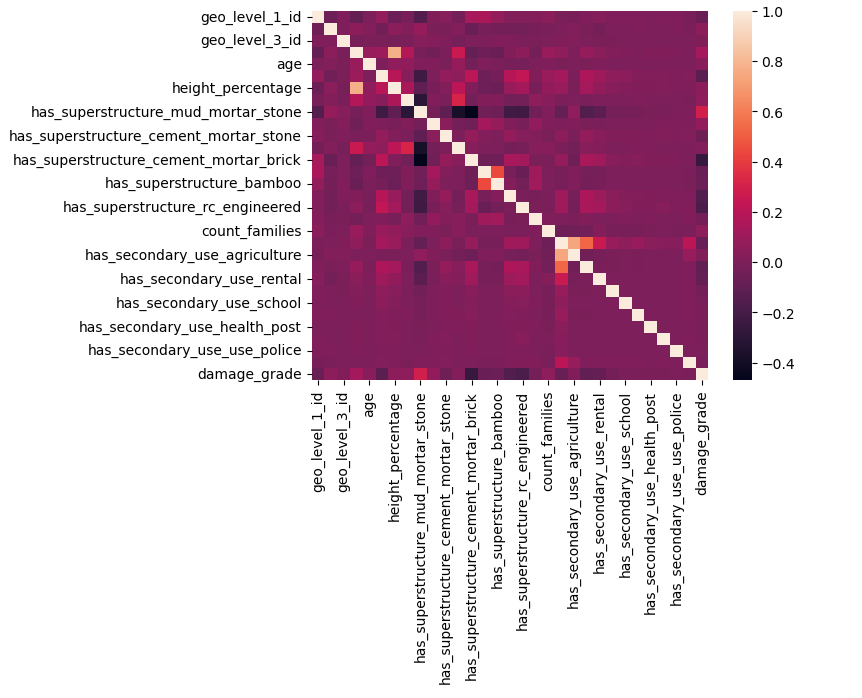
has\_secondary\_use\_health\_post (binary): binanın sağlık direği olarak kullanılıp kullanılmadığını gösterir

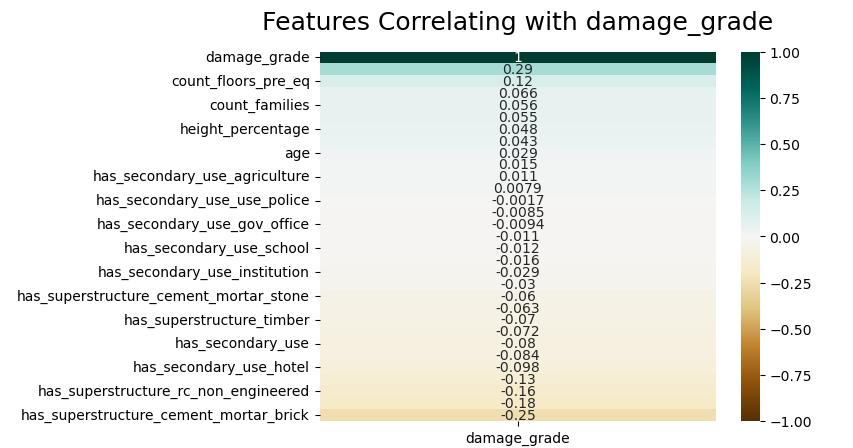
has\_secondary\_use\_gov\_office (binary): binanın bir devlet dairesi olarak kullanılıp kullanılmadığını gösterir

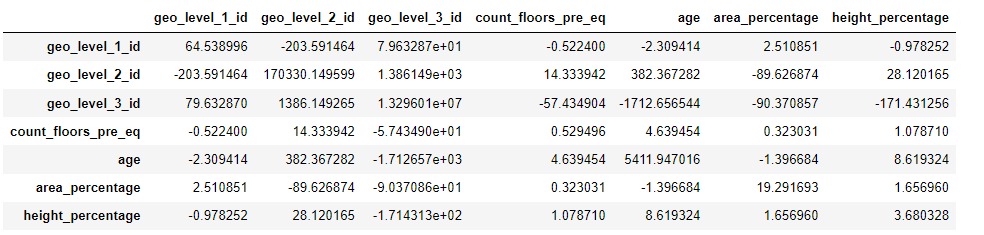
has\_secondary\_use\_use\_police (binary): binanın polis karakolu olarak kullanılıp kullanılmadığını gösterir

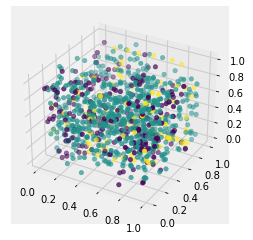
has\_secondary\_use\_other (binary): binanın ikincil olarak başka amaçlar için kullanılıp kullanılmadığını gösterir

## Veri İşleme

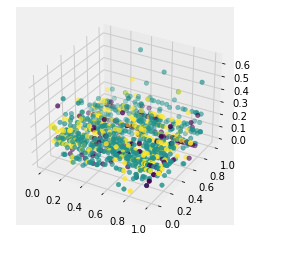
* Kullandığım veri seti önceden işlenmiş ayrıca herhangi bir eksik veri bulunmuyor. Kullandığım veri unstructured bir veri.
* Nominal veriler sıralı değil ayrıca karşılaştırılabilir de değil.
* Öznitelik seçimini sklearn kütüphanesinin SelectKBest algoritması ile yaptım. Annova F-değeri’ne göre en iyi 20 özniteliği seçiyor.
* Sınıf dağılımı 2 numaralı sınıfta yoğunlaştığı için imblearn kütüphanesinden RandomSampler ve SMOTE algoritmalarını kullandım fakat negatif bir etkiyle karşılaştım.
* Data ve Öznitelik normalizasyonunu sklearn kütüphanesinin MinMaxScaler algoritması ile yaptım.
* Veri görselleştirmeleri:
  + Sayısal veriler:
  + Binary veriler:
* Kategorik Veriler:
* Korelasyonlar:



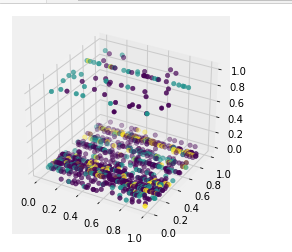
* Kovaryans matrisi:
* Clustering:



geo\_level\_id’ler ile yapılan k-means



Identify applicable funding agency here. If none, delete this text box.

Geo\_level 1,2 ile age özniteliği ile yapılan k-means

Geo\_level 1,2 ile bir binary sınıf arasında k-means

# KULLANILAN MODELLER

## Eğitim için verilerin %80’ini kullandım. Eğitim verilerinin de %20’si ile çapraz doğrulama yaptım (5 fold). En iyi modeli bulduktan sonra o modeli kalan %20 test verisi üzerinde test ettim.

## Problemimiz bir multiclass problemi verileri 3 sınıfa ayırmamız isteniyor.

## KNeighborsClassifier: Bu sınıflandırıcı en yakın k komşunun oylaması ile sınıflandırma yapıyor.

Ben en yakın 3 komşu üzerinden kullandım.

## DecisionTreeClassifier: Karar ağacı non-parametric supervised bir algoritma uzayı gini impurity veya entrophy’I göz önüne alarak çeşitli alt alanlara bölüp buna göre sınıflandırma yapıyor.

max\_features: en iyi ayırmayı ararken kaç özelliğe bakılacağı. None kullanıldı çünkü tüm özelliklere bakılması isteniyor.

max\_depth: ağacın maksimum derinliği. Deneysel olarak 45 verildi.

min\_samples\_split: Bir iç düğümü bölmek için gereken minimum örnek sayısı. 3 seçildi.

min\_samples\_leaf: Bir yaprak düğümde olması gereken minimum numune sayısı. 30 seçildi.

random\_state: Tahmincinin rastgeleliğini kontrol eder. 42 seçildi(42 hayat, evren ve her şeye dair nihai sorunun cevabıdır).

class\_weight: Sınıfların ağırlıklarını belirleyen fonksiyon sınıflarımız dengesiz olduğu için dengelemesi için ‘balanced’ değerini veriyoruz.

## RandomForestClassifier: Random Forest, veri kümesinin çeşitli alt örneklerinde bir dizi karar ağacı sınıflandırıcısı uygulayarak ve over-fitting’i önleyerek sınıflandıran bir yapıdır.

Criterion: Bölünmenin kalitesini ölçme işlevi. Entrophy seçildi.

max\_features: En iyi bölünmeyi ararken göz önünde bulundurulması gereken özelliklerin sayısı. Öznitelik sayısının kökü seçildi.

n\_estimators: Ormandaki ağaç sayısı. 280 seçildi.

class\_weight: Sınıfların ağırlıklarını belirleyen fonksiyon sınıflarımız dengesiz olduğu için dengelemesi için ‘balanced’ değerini veriyoruz.

## AdaBoostClassifier: Bir AdaBoost sınıflandırıcısı, orijinal veri kümesine bir sınıflandırıcı sığdırarak başlayan ve daha sonra sınıflandırıcının ek kopyalarını aynı veri kümesine sığdıran ancak yanlış sınıflandırılan örneklerin ağırlıklarının, sonraki sınıflandırıcıların daha zor durumlara odaklanacağı şekilde ayarlandığı bir meta-tahmin edicidir.

## GradientBoostingClassifier: GradientBoosting, güçlü bir tahmine dayalı model oluşturmak için birçok zayıf öğrenme modelini bir araya getiren bir grup makine öğrenimi algoritmasıdır.

## MLPClassifier: Çok katmanlı bir algılayıcı, tamamen bağlantılı bir ileri beslemeli yapay sinir ağı sınıfıdır.

Solver: Ağırlık optimizasyonu için çözücü. 'lbfgs', yarı-Newton yöntemler ailesindeki bir optimize edicidir.

Max\_iter: Maksimum yineleme sayısı. Çözücü, yakınsama veya bu sayıda yineleme olana kadar yinelenir.

hidden\_layer\_sizes: i. gizli katmandaki nöron sayısı. (5,5) ve (10,10) olarak seçildi.

## XGBClassifier: XGBoost, gradyan destekli ağaç algoritmasının popüler ve verimli bir açık kaynaklı uygulamasıdır.

N\_estimators: The number of trees in the forest. 100 ve 500 denendi.

Subsample: Eğitim örneklerinin alt örnek oranı. 0.8 ve 1.0 seçildi.

Gamma : Ağacın bir yaprak düğümünde daha fazla bölüm oluşturmak için gereken minimum kayıp azaltma. 0, 1 ve 5 seçildi.

max\_depth: ağacın maksimum uzunluğu

learning\_rate: Öğrenme oranı. 0.1 ve 0.3 seçildi.

* Elimdeki bir sınıflandırma problem olduğu için sınıflandırmada daha doğru sonuçlar veren modeler seçilmeye çalışınıldı. Ayrıca bir tane de sinir ağı olan Multi layer perceptron modeli kullanıldı fakat çok derin çalıştırılınmadığı için sonuçları diğerlerinin altında kaldı.

# TEST SONUÇLARI (F1 Bazında

Training ve test yazan değerler accuracy değerleri.

## En iyi KNN Sonucu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## En iyi Decision Tree Classifier Sonucu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## En iyi Random Forrest Classifier Sonucu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## En iyi AdaBoost Classifier Sonucu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## En iyi Gradient Boosting Classifier Sonucu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

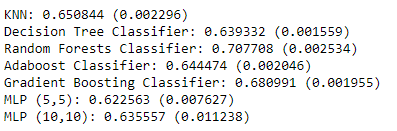
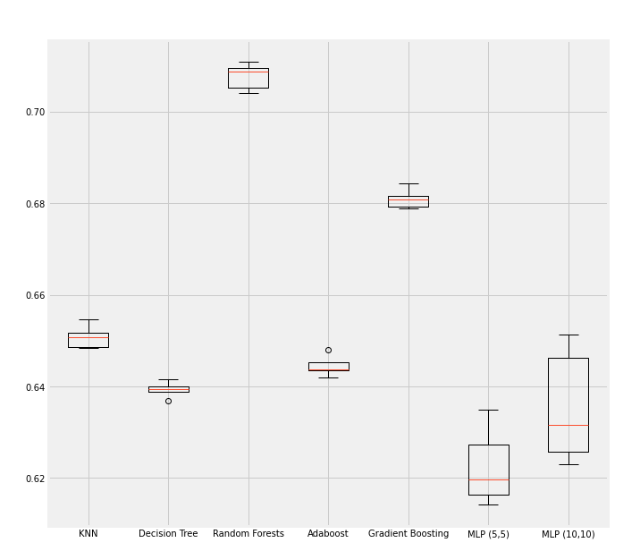
## metin içeren bir resim Açıklama otomatik olarak oluşturulduEn iyi (5,5) MLP Classifier Sonucu

## metin içeren bir resim Açıklama otomatik olarak oluşturulduEn iyi (10,10) MLP Classifier Sonucu

## metin içeren bir resim Açıklama otomatik olarak oluşturulduEn iyi XGBoost Classifier Sonucu

## En iyi Modelin test verisi üzerindeki sonucu

* En iyi XGBoost Sınıflandırıcı çalışıyor. Bunun sebebi problemimizin uzayının çok boyutlu olması ve verimizin aşırı iyi bir şekilde derin öğrenmeye optimize edilmiş olmaması. Feature generation vesaire yapılmadı.
* Aşağıda XGBoost hariç diğer modellerin karşılaştırılması var. Aşağıdaki skorlar accuracy üzerinden hesaplanmıştır.



# SONUÇLAR

* Bu çalışmada ilk önce verinin incelemesini yaptık. Daha sonra dağılımını kontrol ettik ve dağılımının düzgün olmadığını gördüğümüz için sınıflandırma methodlarına sınıfları ağırlıkları dengeli olacak şekilde verdik. Modellere vermeden önce kategorik veriler arasında bir büyüklük ilişkisi göremediğimiz için one hot encoding ile preprocessing yaptık. Daha sonra modellerin hesaplama hızlarını arttırmak ve local extremumlardan etkilenmemek için normalizasyon yaptık. Öznitelik çıkarımı yapmayı denedik fakat sonuçları negatif etkilediği için kullanmadık. Daha sonra kfold kullanarak %60 train %20 validation ve %20 test olmak üzere eğitimlerimizi gerçekleştirip testlerimizi yaptık.
* Çalışma sonucunda bir sınıflandırma problemine nasıl yaklaşılması gerektiğini, özniteliklere nasıl yaklaşılınacağını ve eğitilen modelleri en verimli şekilde nasıl kullanacağımızı gördük. Kazancımız ise çok fazla öznitelik çıkarımı vesaire yapmadan zar atmaktan daha iyi bir sonuca ulaşabileceğimizi gördük.
* Çalışmada yapmadığımız yapay veri oluşturma vey apay

##### Appendix

Random Forest’ın kullandığı en önemli öznitelikler

