EGE ÜNİVERSİTESİ

EÜ MÜHENDİSLİK FAKÜLTESİ

LİSANS TEZİ

BÜYÜK VERİNİN VERİ ANALİZİ TEKNİKLERİ İLE MANİPÜLE EDİLİP HAZIRLANMASI VE MAKİNE ÖĞRENMESİ MODELLERİNİN EĞİTİLMESİNDE KULLANILMASI

Berat DEMİR

Tez Danışmanı : Prof. Dr. Vecdi AYTAÇ

Bilgisayar Mühendisliği Bölümü

Sunuş Tarihi : 20.01.2022

Bornova - İZMİR
2022

EGE ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ

(LİSANS TEZİ)

BÜYÜK VERİNİN VERİ ANALİZİ TEKNİKLERİ İLE MANİPÜLE EDİLİP HAZIRLANMASI VE MAKİNE ÖĞRENMESİ MODELLERİNİN EĞİTİLMESİNDE KULLANILMASI

Berat DEMİR

Tez Danışmanı : Prof. Dr. Vecdi AYTAÇ

Bilgisayar Mühendisliği Bölümü

Sunuş Tarihi : 20.01.2022

Bornova - İZMİR
2022

Berat DEMİR ve tarafından **Lisans** tezi olarak sunulan **"BÜYÜK VERİNİN VERİ ANALİZİ TEKNİKLERİ İLE MANİPÜLE EDİLİP HAZIRLANMASI VE MAKİNE ÖĞRENMESİ MODELLERİNİN EĞİTİLMESİNDE KULLANILMASI"** başlıklı bu çalışma EÜ Lisans Eğitim ve Öğretim Yönetmeliği ile EÜ Mühendislik Fakültesi Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve **20.01.2022** tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri :		**İmza**
Jüri Başkanı	**:** Prof. Dr. Vecdi AYTAÇ	.....
Raportör Üye	**:** Prof. Dr. Öğretim Üyesi - 1	.....
Üye	**:** Doç. Dr. Öğretim Üyesi - 2	.....

EGE ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisans Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Lisans Tezi olarak sunduğum **"BÜYÜK VERİNİN VERİ ANALİZİ TEKNİKLERİ İLE MANİPÜLE EDİLİP HAZIRLANMASI VE MAKİNE ÖĞRENMESİ MODELLERİNİN EĞİTİLMESİNDE KULLANILMASI"** başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

20.01.2022

Berat DEMİR

ÖZET

BÜYÜK VERİNİN VERİ ANALİZİ TEKNİKLERİ İLE MANİPÜLE EDİLİP HAZIRLANMASI VE MAKİNE ÖĞRENMESİ MODELLERİNİN EĞİTİLMESİNDE KULLANILMASI

DEMİR, Berat

Lisans Tezi, Bilgisayar Mühendisliği Bölümü
Tez Danışmanı: Prof. Dr. Vecdi AYTAÇ
Ocak 2022, 48 sayfa

Bu tezde büyük verinin, veri analizi teknikleri kullanılarak verilerin analiz edilmesi ve makine öğrenmesi algoritmalarının tahminlemesinde kullanılması hedeflenmiştir. Tez konusunun verimli bir şekilde hayata geçirilebilmesi için hazırlanan verilerin, birden fazla makine öğrenmesi algoritmasının eğitilmesinde ve test edilmesinde kullanılmıştır.Veri analizi; veri temizleme, eksik verileri doldurma, öznitelik mühendisliği, sayısal veri tiplerini ölçeklendirme, kategorik veri tiplerini enkode etme, gibi alt bölümleri içermektedir. Bu hedefin gerçekleştirilmesi için uygulama gerçekleştirilmiştir. Bu uygulama ile kesikli değişkenlere sahip verilerin(kolonları) kategorik verilere(kolonlar) dönüştürümüştür. Bu dönüştürme sonucunda elde edilen veriseti ile makine öğrenmesi algoritmaları eğitilmiştir. Eğitilen modellerin performansları ölçülmüştür. Tezin uygulaması Python programla dili ve bu dile ait kütüphaneler(Numpy, Pandas, Matplotlib, Scikitlearn) kullanılarak Jupyter Notebook ortamında gerçekleştirilmiştir.

Anahtar sözcükler: Python, Numpy, Pandas, Matplotlib, Scikitlearn, Jupyter Notebook, Train Test Split, StratifiedShuffleSplit, OneHotEncoder, BaseEstimator TransformerMixin, Pipeline, LinearRegression, DecisionTreeRegressor, RandomForestRegressor, Cross Val Score, GridSearchCV, R2 Score

ABSTRACT

MANIPULATION AND PREPARATION OF BIG DATA WITH DATA ANALYSIS TECHNIQUES AND TRAINING MACHINE LEARNING MODELS

DEMİR, Berat

BSc in Computer Engineering Department
Supervisor: Prof. Dr. Vecdi AYTAÇ
January 2022, 48 pages

In this thesis, data were analyzed using data analysis techniques. It is aimed to be used in the estimation of machine learning algorithms trained with the analyzed data. In order to implement the thesis topic efficiently, it has been used in training and testing more than one machine learning algorithm with the analyzed data. Data analysis includes subsections such as data cleaning, filling in missing data, feature engineering, scaling numerical data types. With this application, data (columns) with discrete variables are transformed into categorical data (columns). Machine learning algorithms are trained with the data set obtained as a result of this transformation. The performances of the trained models were measured.The application of the thesis was implemented in the Jupyter Notebook environment by using the Python programming language and its libraries (Numpy, Pandas, Matplotlib, Scikitlearn)

Keywords: Python, Numpy, Pandas, Matplotlib, Scikitlearn, Jupyter Notebook, Train Test Split, StratifiedShuffleSplit, OneHotEncoder, BaseEstimator TransformerMixin, Pipeline, LinearRegression, DecisionTreeRegressor, RandomForestRegressor, Cross Val Score, GridSearchCV, R2 Score

TEŞEKKÜR

Bu çalışma sırasında yardımlarını ve bilgeliğini bizden asla esirgemeyen tez danışmanımız Prof. Dr. Vecdi Aytaç'a ve teknik bilgilerini paylaşan meslektaşlarımıza en içten bir teşekkürü borç biliriz.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

LİST OF TABLES

ALGORİTMALAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
ML	Machine Learning

1. GİRİŞ

Bu tez çalışmasının odaklandığı konu veri analizi sürecinin nasıl işlediğinin kavranmasıdır. Projenin konusu veri analizi teknikleri ile büyük veriyi analiz etmek, analiz edilen verinin ML modellerinin eğitilmesinde ve yapacağı tahminlerde kullanılmasıdır. Projenin amacı veri bilimine giriş yapmak ve bu alanda uygulama geliştirerek veri analizi ve ML konularında beceri kazanmaktır. Geliştirilen uygulama ile verilerden daha anlamlı veriler elde etmek ve elde dilen verilerin ML modellerine katgıda bulunup ya da bulunmadığını ölçmek bu tezin en temel amacıdır. Yan amaçlar ise hazır kütüphane kullanımını minimum seviyede tutmak, emek harcayarak veri analizi sürecini otomatize eden kütüphane geliştirmek, numerik veri tipindeki kolonlar kesikli değişkene sahip ise bu kolonları kategorik veri tipine dönüştürerek ML modellerinin performansını ölçmek ve farklı ölçeklendirme methodlarıyla ölçeklendirilen verilerin ML modellerindeki performanslarını ölçmektir. Tezin araştırma konuları arasında aşağıdaki sorular yer almaktadır;

Tasarlanan projede hangi teknolojiler kullanılacaktır? Hedeflenen uygulamanın hazır kütüphanelerden farkı neler olacak? Mevcut verilerden daha anlamlı veriler üretilebilinir mi? Eldeki tüm veriler ML modelleri eğitildiğinde daha iyi tahminde bulunur mu? Korelasyon katsayısı belli bir eşik değerinin üstündeki verilerle eğitilen ML modelleri daha iyi performans verir mi? Bu tezin, veri bilimine ilgili olan herkese verilerin nasıl analiz edildiği konusunda oldukça yararı olacaktır.

Uygulama başarılı bir şekilde belirli bir seviyeye ulaştıktan sonra uygulamadaki özellikler daha da geliştirilebilir örnek: öznitelik mühendisliği bileşeniyle eldeki verilerle daha kapsamlı hesaplamalar yaparak daha anlamlı veriler üretilebilir.

2. YAPILAN ÇALIŞMALAR

Bu kısımda tezin tasarlanma aşamasından gerçekleştirilmesine kadar tüm kısımlar detaylı olarak anlatılmaktadır. Araştırılan konu senelerdir bilgisayar bilimlerinde araştırılmakta olup, birçok farklı yaklaşım ve birçok farklı prototip üzerinde çalışılmıştır. Aşağıdaki başlıklarda projeye yakın olan uygulamaları, kullanılan teknolojileri ve bu projede nasıl ilerlendiğini bulacaksınız.

2.1 Önceki Çalışmalar

2.1.1 Metodolojiye Yaklaşımlar

Öznitelik mühendisliği, makine öğrenimi için veri hazırlamada merkezi bir görevdir. Gelişmiş tahmin performansına yol açan belirli özelliklerden uygun özellikler oluşturma uygulamasıdır. Öznitelik mühendisliği, yenilerini oluşturmak için verilen özellikler üzerinde aritmetik ve toplama operatörleri gibi dönüşüm fonksiyonlarının uygulanmasını içerir. (Mamur et al., 2017)

Normalleştirme, verileri 0 ile 1 veya -1 ile +1 gibi belirli bir aralığa dönüştürme işlemi. Farklı özelliklerin aralıklarında büyük farklılıklar olduğunda normalleştirme gereklidir. Bu ölçekleme yöntemi, veri kümesi aykırı değerler içermediğinde kullanışlıdır. Z-skor standardizasyonu, veri setini Ortalaması = 0 ve standart sapması = 1 olan veri setine dönüştürür. Bu ölçekleme yöntemi, veriler normal dağılıma (Gauss dağılımına) sahipse kullanışlıdır, eğer veriler normal dağılıma sahip değilse sorun çıkaracaktır.(Ali and Faraj, 2014)

2.2 Kullanılan Teknolojiler

2.2.1 Python Programlama Dili

Python, yorumlanmış üst düzey genel amaçlı bir programlama dilidir. Tasarım felsefesi, önemli girinti kullanımıyla kod okunabilirliğini vurgular. Dil yapıları ve nesne yönelimli yaklaşımı, programcıların küçük ve büyük ölçekli projeler için temiz kodlar yazmasına yardımcı olmayı amaçlar. Python Yapılandırılmış (özellikle prosedürel), nesne yönelimli ve işlevsel programlama dahil olmak üzere çoklu programlama paradigmalarını destekler.(van Rossum, 1991)

2.2.2 Numpy

NumPy, Python programlama dili için büyük, çok boyutlu dizileri ve matrisleri destekleyen, bu diziler üzerinde çalışacak üst düzey matematiksel işlevler sağlan bir kütüphanedir.(Oliphant, 2016)

2.2.3 Pandas

Pandas, veri işlemesi ve analizi için Python programlama dilinde yazılmış olan bir yazılım kütüphanesidir. Bu kütüphane temel olarak zaman etiketli serileri ve sayısal tabloları işlemek için bir veri yapısı oluşturur ve bu şekilde çeşitli işlemler bu veri yapısı üzerinde gerçekleştirilebilir.(McKinney, 2008)

2.2.4 Matplotlib

Matplotlib, Python programlama dili ve sayısal matematik uzantısı NumPy için bir çizim kütüphanesidir. Tkinter, wxPython, Qt veya GTK gibi genel amaçlı GUI araç takımlarını kullanarak grafikleri uygulamalara gömmek için nesne yönelimli bir API sağlar.(Hunter, 2003)

2.2.5 Scikit-learn

Scikit-learn (eski adıyla scikits.learn ve sklearn olarak da bilinir), Python programlama dili için ücretsiz bir makine öğrenimi kütüphanesidir.Destek vektör makineleri, rastgele ormanlar, gradyan artırma, k-ortalamalar ve DBSCAN dahil olmak üzere çeşitli sınıflandırma, regresyon ve kümeleme algoritmalarına sahiptir.(Vikipedi, 2007)

- **Train Test Split :** Veri setini, random bir şekilde train set, test set olarak ayırır.
- **Stratified Shuffle Split :** Verileri train/test setlerine bölmek için train/test indekslerini sağlar.
- **One Hot Encoder :** Kategorik verileri numerik dizilere dönüştürür
- **Pipeline :** Aamacı, farklı parametreleri ayarlarken birlikte çapraz doğrulanabilen birkaç adımı bir araya getirmektir.
- **Cross Val :** Cross-validation, makine öğrenmesi modelinin görmediği veriler üzerindeki performansını mümkün olduğunca objektif ve doğru

bir şekilde değerlendirmek için kullanılan istatistiksel bir yeniden örnekleme(resampling) yöntemidir. İkinci bir kullanım alanı ise modelde hiperparametre optimizasyonu yapmaktır.(Durna, 2020a)

- **Grid Search CV :** Modelde denenmesi istenen hiperparametreler ve değerleri için bütün kombinasyonlar ile ayrı ayrı model kurulur ve belirtilen metriğe göre en başarılı hiperparametre seti belirlenir.(Durna, 2020b)
- **R2 Score :** Verilerin yerleştirilmiş regresyon hattına ne kadar yakın olduğunun istatistiksel bir ölçüsüdür. Ayrıca belirleme katsayısı veya çoklu regresyon için çoklu belirleme katsayısı olarak da bilinir. Daha basit bir dilde söylemek gerekirse R-kare, doğrusal regresyon modelleri için uygunluk ölçüsüdür.(Şevket Ay, 2020)

2.2.6 Linear Regression

Linear Regression, veri kümesinde gözlemlenen hedefler ile doğrusal yaklaşım tarafından tahmin edilen hedefler arasındaki kalan karelerin toplamını en aza indirmek için w = (w1, …, wp) katsayılarına sahip doğrusal bir modele uyar.

Algoritma 1: Linear Regression

Data: Prepared Data
Result: Parameters

1 Start
2 Read Number of Data (n)
3 For i =1 to n: Read Xi and Yi Next i
4 Initialize: sumX = 0 sumX2 = 0 sumY = 0 sumXY = 0
5 Calculate Required Sum For i =1 to n: sumX = sumX + Xi sumX2 = sumX2 + Xi * Xi sumY = sumY + Yi sumXY = sumXY + Xi * Yi Next i
6 Calculate Required Constant a and b of y = a + bx: b = (n * sumXY - sumX * sumY)/(n*sumX2 - sumX * sumX) a = (sumY - b*sumX)/n
7 Display value of a and b
8 Stop

2.2.7 Decision Tree Regressor

Bir karar ağacı, bir dizi olası karar yolunu ve her bir yol için bir sonucu temsil etmek için bir ağaç yapısı kullanır.(Grus, 2019)

Algoritma 2: Decision Tree

Data: Training Set
Result: Target Feature
TreeGrowing (S,A,y)
Where:
S - Training Set
A - Input Feature Set
y - Target Feature
Create a new tree T with a single root node
IF One of the Stopping Criteria is fulfilled THEN:
Mark the root node in T as a leaf with the most
common value of y in S as a label.
ELSE :
Find a discrete function f(A) of the input
attributes values such that splitting S
according to f(A)'s outcomes (v1,...,vn) gains
the best splitting metric.
IF best splitting metric > treshold **THEN:**
Label t with f(A)
FOR each outcome vi of f(A):
Set Subtreei = TreeGrowing (f(A) =viS,A,y).
Connect the root node of tT to Subtreei with
an edge that is labelled as vi
END FOR
ELSE
Mark the root node in T as a leaf with the most
common value of y in S as a label.
END IF
END IF
RETURN T
TreePruning (S,T,y)
Where:
S - Training Set
y - Target Feature
T - The tree to be pruned
DO:
Select a node t in T such that pruning it
maximally improve some evaluation criteria
IF t6 =Ø THEN T =pruned(T,t)
UNTIL t =Ø
RETURN T

2.2.8 Random Forest Regressor

Rastgele Orman Regresyonu, regresyon için topluluk öğrenme yöntemini kullanan denetimli bir öğrenme algoritmasıdır. Topluluk öğrenme yöntemi, tek bir modelden daha doğru bir tahmin yapmak için birden çok makine öğrenimi algoritmasından gelen tahminleri birleştiren bir tekniktir.(Bakshi, 2020)

Algoritma 3: Random Forest Regression

Data: Training Set
Result: The learned tree
Precondition: A training set S := (x1, y1), . . . ,(xn, yn), features F, and number
of trees in forest B.
function RandomForest(S , F)
H $\longleftarrow 0$
for i $\epsilon 1, ..., Bdo$
$S^i \longleftarrow Abootstrapsample from S$
h(i) $\longleftarrow RandomizedTreeLearn(S^i, F)$
H $\longleftarrow H \cap hi$
end for
return H
end function
function RandomizedTreeLearn(S , F)
At each node:
f $\longleftarrow verysmallsubset of F$
Split on best feature in f
return The learned tree
end function

2.3 Projenin Gerçekleştilmesi

Projeye başlamadan önce kapsamlı bir literatür çalışması yürütülmüştür. Eğitim videoları ve kitapları incelenmiştir. Proje konusu aşamaları belli bir sıraya göre olduğundan herhangi bir zorluğu omamıştır. Projede kullanılan veri seti kaggle.com web sitesinden indirilmiştir. Veri seti evin özelliklerini ve satış fiyatını içeriyor. Veri seti 79 tane öznitelik ve satış fiyatıyla birlikte toplam 80 tane kolona sahiptir. Projeyi meydana getiren sistemin bileşenleri kapsamlı bir şekilde aşağıda çıklanarak listelenmiştir.

2.3.1 Eksik Verileri Doldurma

Aşağıdaki kod ilk aşamada veri setinde sutunlardaki eksik hücrelerin sayısı %50'nin üzerindeyse o sutunlar veri setinden atılır. İkinci aşamada veri tipi numerik olan sutunlarda eksik olan hücreler bulunduğu sutunun median değeriyle doldurulur. Üçüncü aşamada veri tipi kadegorik olan sutunlarda ise eksik olan hücreler bulunduğu sutunda frekansı yüksek olan verilerle doldurulur.

```
class Fill_Miss_Value(BaseEstimator, TransformerMixin):

    def __init__(self, drop_con=False):
        self.drop_con = drop_con

    def fit(self, X, y=None):

        global  numerical_att, categorical_att, frequency_cat,
                droped_colum, median

        droped_colum = []

        #drops columns that less than half length values
        if (self.drop_con):
            a = np.ceil(len(X)*(0.5)),
            for att in list(X):
                if X[att].notnull().sum() < a:
                    droped_colum.append(att)
                    X.drop(att, axis=1, inplace=True)

        num_att=[att for att in dict(Train_set.dtypes)
                     if dict(Train_set.dtypes)[att] != 'object']
        frequency_cat = {}

        categorical_att= X.columns.symmetric_difference(num_att)

        median = X[num_att].median(axis=0)

        for att in categorical_att:
            frequency_cat[att] = X[att].value_counts()
        return self

    def transform(self, X):
        median = X[num_att].median(axis=0)

        for att, val in zip(num_att, median):
            X[att].fillna(val, inplace=True)

        for key , val in frequency_cat.items():

            X[key].fillna(val.idxmax(), inplace=True)

            categorical_att = list(X)
        return X
```

Listing 2..1. Eksik verileri doldurma

2.3.2 Öznitelik Çıkarsama

Veri setindeki bazı veriler çok anlamlı olmayabilir bu yüzden aşağıdaki sistem bileşeni olan kod parçası daha anlamlı veriler üretmek için tasarlanmıştır. Bu bileşenin fonksiyonu daha anlamlı veriler üretmek için iki numerik veri üzerinde toplama, çıkarma, çarpma ve bölme gibi işlemler yaparak yeni veriler üretebilmektedir, kesikli değişkene sahip verileri kategorik verilere döştürebilmektedir.

```
class Feature_Combine_Attribute_Adder(BaseEstimator,
                                      TransformerMixin):

    def __init__(self, operation_atts=None, drop_columns=None,
                 drop_con=False):
        self.drop_columns = drop_columns
        self.operation_atts = operation_atts
        self.drop_con = drop_con

    def fit(self, X, y=None):
        return self



    def transform(self, X):

       #select operator for feature operation
        operation = {'+': (lambda a, b: a + b),
                     '-': (lambda a, b: a - b),
                     '/': (lambda a, b: a / b),
                     '*': (lambda a, b: a * b)}


        for key , val in self.operation_atts.items():

            if isinstance(val, str):

                if (len(val.split()) == 1) :

                    #numerical attribute transform
                    #to catecorical attribute
                    if ((X.dtypes)[val] in ['int32', 'int64',
                                            'float32', 'float64']):

                        mapping = dict(zip(X[val].unique(),
                               [str(m) for m in X[val].unique()]))

                        X[key] = X[val].map(mapping)


                if (len(val.split()) >= 2):

                    #both of them numerical attribute for combine
```

```
                att1, opp, att2 = val.split()

                X[key] = operation[opp](X.loc[:,att1].values,
                                        X.loc[:,att2].values)


            #numerical attribute transform to categorical
            else:
                new_att_name = key + '_' + 'CAT'

                X[new_att_name] = pd.cut(X[key],
                                         len(val), labels=val)
                X[new_att_name] = X[new_att_name].astype('object')

        if (self.drop_con):

            return X.drop(self.drop_columns, axis=1, inplace=True)

        return X
```

Listing 2..2. Feature Engineering

2.3.3 Kategorik Verileri Numerik Verilere Dönüştürme

Makine öğrenmesi algoritmaları sayısal verilerle çalışır bu yüzden veri setindeki kategorik veriler One Hot Encoder ile sayısal verilere dönüştürülmelidir. Bu yüzden aşağıdaki kod parçası ile bahsi edilen işlemler gerçekleştirilir.

```
class TextColum_To_CategoricColum(TransformerMixin):

    def __init__(self):
        self

    def fit(self, X, y=None):
        return self

    def transform(self, X):

        categorical_att = [att for att in dict(X.dtypes)
                           if dict(X.dtypes)[att] in ['object']]


        oneHot=OneHotEncoder(sparse=False).fit(X[categorical_att])
        oneHot_data = oneHot.transform(X[categorical_att])
        oneHot_columns=[name for name in
                        oneHot.get_feature_names(categorical_att)]

        df = pd.DataFrame(oneHot_data, index=X.index,
                          columns=oneHot_columns, dtype=np.int32)

        return pd.concat([X.drop(categorical_att, axis=1), df],
                                 axis=1)
```

Listing 2..3. Transform Categoric Columns To OneHotEncode

2.3.4 Sayısal Verileri Ölçeklendirme

Makine öğrenmesi algoritmaları eğitilmeden önce verilerin ölçeklendirilmesi gerekir çünkü veriler ölçeklendirilmediği zaman ML algoritmalrı sağlıklı tahminde bulunmayabilirler bu yüzden aşağıdaki kod parçası iki farklı ölçeklendirmeye sahiptir. İlki standard ölçekleyici diğeri normalize ölçekleyicidir, amaç verileri her iki ölçekleyiciyle ölçekelendirerek ML algoritmalarını eğitmek.

```
class Scaler(BaseEstimator, TransformerMixin):

    def __init__(self, min_max=False):
        self.min_max = min_max

    def fit(self, X, y=None):

        return self

    def transform(self, X):

        numerical_att=[att for att in dict(Train_set.dtypes)
                if not(dict(Train_set.dtypes)[att] in ['object'])]

        if (self.min_max):
            scaler = MinMaxScaler()
            data = scaler.fit_transform(X[numerical_att])
            X[numerical_att]= pd.DataFrame(data, index=X.index,
                              columns=numerical_att)[numerical_att]
        else:
            scaler = StandardScaler()
            data = scaler.fit_transform(X[numerical_att])
            X[numerical_att]= pd.DataFrame(data, index=X.index,
                              columns=numerical_att)[numerical_att]

        return X
```

Listing 2..4. Scaling Numeric Data

2.3.5 Kolon Sayısını Azaltma

Bazen veri setindeki verilerin birbirleriyle arasındaki ilişki yok denecek kadar azdır ya da hiç yoktur, bu yüzden ML algoritmalarını daha arındırlmış verilerle eğitmek amaçlanmıştır. Bu yüzden aşağıdaki kod parçası belirlenecek olan eşik değerinin altındaki korelasyon katsayısına sahip sutunları veri setinden atacaktır.

```
class Column_Reduction(BaseEstimator, TransformerMixin):

    def __init__(self, Label=None, thresh=0.0,  drop_con=False,
```

```
                drp_list=None, train_con=False):
        self.thresh = thresh
        self.Label = Label
        self.drop_con = drop_con
        self.drp_list = drp_list
        self.train_con = train_con

    def fit(self, X, y=None):
        return self

    def transform(self, X):

        if (self.drop_con):
            X.drop(self.drp_list, axis=1, inplace=True)

        if (self.train_con):
            X[self.Label.name] = self.Label

            corr_matrix = X.corr()

            temp_list = corr_matrix[self.Label.name]
            [(np.abs(corr_matrix[self.Label.name])>=self.thresh)&
                (np.abs(corr_matrix[self.Label.name])<1.0)].index

            return X[temp_list]
```

Listing 2..5. Column Reduction

2.3.6 Veri Setini Train-Test Setine Ayırma

Aşağıdaki kod parçası veri setini train ve test setine ayırır, bu ayırma işlemini biraz detaylandırırsak daha faydalı olacaktır çünkü ayırma işleminde test setinin veri setini temsil etmesi gerekmektedir.

```
class Split_Data_To_Train_Test_Set(BaseEstimator,
                                   TransformerMixin):

    def __init__(self, Label=None,test_size=0.2):

        self.Label = Label
        self.test_size = test_size

    def fit(self, X, y=None):

        return self

    def transform(self, X):

        X.drop([self.Label.name], axis=1, inplace=True)

        split = StratifiedShuffleSplit(n_splits=1,
                    test_size=self.test_size, random_state=42)
```

```
        X[self.Label.name] = self.Label.values

        for train_index, test_index in split.split(X,
                                                   self.Label.values):
            train_set = X.iloc[(train_index - 1)]
            test_set = X.iloc[(test_index - 1)]



        return (train_set, test_set)
```

Listing 2..6. Column Reduction

2.3.7 Verileri Yükleme

kaggle.com web sitesinden indirilen verilerle çalışmak için öncelikle veriler yüklenmeli. Veriler yüklendikten sonra veriler üzerinde çalışırken orjinal verilerin değişmemesi için veri setinin kopyası oluşturlmalıdır, bu yüzden aşağıdaki kod parçası bahsedilen işlemleri gerçekleştirir.

```
Test_set = pd.read_csv("house-prices-advanced
                        -regression-techniques/test.csv")

Test_set_Labels = pd.read_csv("house-prices-advanced
           -regression-techniques/sample_submission.csv")

Train_set = pd.read_csv("house-prices-advanced
                       -regression-techniques/train.csv")

#Train_set.drop(Train_set[(Train_set['SalePrice'] >
                            400000)].index, inplace=True)


Train_set_Labels = Train_set['SalePrice']


Train_set_Labels.index = Train_set['Id'].values

Train_set.drop(['SalePrice'], axis=1, inplace=True)

for set_ in (Test_set, Test_set_Labels, Train_set):
    set_.index = set_["Id"].values
    set_.drop(['Id'], axis=1, inplace=True)
```

Listing 2..7. Load Data

2.3.8 Veri Setine Hızlı Bakış

Öncelikle veri setindeki veri tipleri ve eksik veriler hızlı bir şekilde incelenirse. Veri seti 1460 tane satır 80 tane sutuna sahiptir, dört tane sutunun neredeyse tamamı

boş ve bu yüzden bu veri setini bu sutunlardan kurtarmak faydalı olacaktır. Geri kalan her sutunda en fazla 259 tane eksik veri var.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1460 entries, 1 to 1460
Data columns (total 79 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   MSSubClass      1460 non-null   int64
 1   MSZoning        1460 non-null   object
 2   LotFrontage     1201 non-null   float64
 3   LotArea         1460 non-null   int64
 4   Street          1460 non-null   object
 5   Alley           91 non-null     object
 6   LotShape        1460 non-null   object
 7   LandContour     1460 non-null   object
 8   Utilities       1460 non-null   object
 9   LotConfig       1460 non-null   object
 10  LandSlope       1460 non-null   object
 11  Neighborhood    1460 non-null   object
 12  Condition1      1460 non-null   object
 13  Condition2      1460 non-null   object
 14  BldgType        1460 non-null   object
 15  HouseStyle      1460 non-null   object
 16  OverallQual     1460 non-null   int64
 17  OverallCond     1460 non-null   int64
 18  YearBuilt       1460 non-null   int64
 19  YearRemodAdd    1460 non-null   int64
 20  RoofStyle       1460 non-null   object
 21  RoofMatl        1460 non-null   object
 22  Exterior1st     1460 non-null   object
 23  Exterior2nd     1460 non-null   object
 24  MasVnrType      1452 non-null   object
 25  MasVnrArea      1452 non-null   float64
 26  ExterQual       1460 non-null   object
 27  ExterCond       1460 non-null   object
 28  Foundation      1460 non-null   object
 29  BsmtQual        1423 non-null   object
 30  BsmtCond        1423 non-null   object
 31  BsmtExposure    1422 non-null   object
 32  BsmtFinType1    1423 non-null   object
 33  BsmtFinSF1      1460 non-null   int64
 34  BsmtFinType2    1422 non-null   object
 35  BsmtFinSF2      1460 non-null   int64
 36  BsmtUnfSF       1460 non-null   int64
 37  TotalBsmtSF     1460 non-null   int64
 38  Heating         1460 non-null   object
 39  HeatingQC       1460 non-null   object
 40  CentralAir      1460 non-null   object
 41  Electrical      1459 non-null   object
 42  1stFlrSF        1460 non-null   int64
 43  2ndFlrSF        1460 non-null   int64
 44  LowQualFinSF    1460 non-null   int64
 45  GrLivArea       1460 non-null   int64
 46  BsmtFullBath    1460 non-null   int64
```

```
 47  BsmtHalfBath    1460 non-null   int64
 48  FullBath        1460 non-null   int64
 49  HalfBath        1460 non-null   int64
 50  BedroomAbvGr    1460 non-null   int64
 51  KitchenAbvGr    1460 non-null   int64
 52  KitchenQual     1460 non-null   object
 53  TotRmsAbvGrd    1460 non-null   int64
 54  Functional      1460 non-null   object
 55  Fireplaces      1460 non-null   int64
 56  FireplaceQu     770 non-null    object
 57  GarageType      1379 non-null   object
 58  GarageYrBlt     1379 non-null   float64
 59  GarageFinish    1379 non-null   object
 60  GarageCars      1460 non-null   int64
 61  GarageArea      1460 non-null   int64
 62  GarageQual      1379 non-null   object
 63  GarageCond      1379 non-null   object
 64  PavedDrive      1460 non-null   object
 65  WoodDeckSF      1460 non-null   int64
 66  OpenPorchSF     1460 non-null   int64
 67  EnclosedPorch   1460 non-null   int64
 68  3SsnPorch       1460 non-null   int64
 69  ScreenPorch     1460 non-null   int64
 70  PoolArea        1460 non-null   int64
 71  PoolQC          7 non-null      object
 72  Fence           281 non-null    object
 73  MiscFeature     54 non-null     object
 74  MiscVal         1460 non-null   int64
 75  MoSold          1460 non-null   int64
 76  YrSold          1460 non-null   int64
 77  SaleType        1460 non-null   object
 78  SaleCondition   1460 non-null   object
dtypes: float64(3), int64(33), object(43)
memory usage: 912.5+ KB
```

Listing 2..8. Veri setine bakış

2.3.9 Öznitelik Mühendisliği

Şekil 2.1.'deki tablodan çıkarsama yapılacak olursa, ilk olarak burdaki verilerin neyi temsil ettiğinden bahsetmek gerekecek.

Yrsold, evin satış tarihini temsil ediyor, **YearBuilt**, evin yapılış tarihini temsil adiyor, Bu iki veriden faydalanarak evin yaşını buluruz bu daha anlamlı bir veri olacaktır.

TotRmsAbvGrd, evin toplam oda sayısını temsil ediyor, **Fireplaces**, evdeki toplam şömine miktarını temsil ediyor, Bu iki veriden faydalanarak bir odaya kaç tane şömine düştüğünü hesaplarsak buda anlamlı bir veri olacaktır, ancak bu iki veri de tek başlarına yeteri kadar anlamlı verilerdir.

MSSubClass, evin sınıfını temsil ediyor, ancak bu kolon kesikli değişkene sahiptir çünkü bu kolundaki birbirinden farklı veri sayısı 15 tir, burdan yola çıkarak bu veri tipini kategorik veri tipine dönüştürerek elde edilen veri setiyle ML algoritmaları eğitilerek algoritmalara katkısını gözlemlemek amaçlanmıştır.

	YrSold	YearBuilt	Fireplaces	TotRmsAbvGrd	MSSubClass
1	2008	2003	0	8	60
2	2007	1976	1	6	20
3	2008	2001	1	6	60
4	2006	1915	1	7	70
5	2008	2000	1	9	60
...	...	...	...	...	...
1456	2007	1999	1	7	60
1457	2010	1978	2	7	20
1458	2010	1941	2	9	70
1459	2010	1950	0	5	20
1460	2008	1965	0	6	20

1460 rows × 5 columns

```
len(Train_set['MSSubClass'].unique())
```

```
15
```

Şekil 2.1. Veri incelene

Yukarıda bahsedilen öznitelik çıkarma işleminin açıklayıcı olması adına detaylı anlatılmıştır, aşağıdaki kod parçası yukarıda anlatılan öznitelik çıkarma işleminin tümünü kapsıyor. Aşağıdaki listede çıkarılan özniteliklerin her biri kısaca özetlenmiştir.

House_Age : Evin yaşı
AfterRemodAge : Ev kaç yıl önce restore edildi
TotalFloorSquareF : Evin toplam yaşam alanı
TotalBsmtFinSF : Evin borumunun toplam alanı
TotalBsmtBath : Evin bodrumundaki yatak odası sayısı

TotalBathroom : Evin yatak odası sayısı bodrumdakiler hariç
Garage_Age : Garajın yaşı
Per_Fireplaces_Room : Oda başına düşen şömine adedi
Cat_MSSubClass : Evin sınıf tipini kategorik tipe dönüştürme

```
arithmetic_operation1 = {
        'House_Age':'YrSold - YearBuilt',
        'AfterRemodAge':'YrSold - YearRemodAdd',
        'TotalFloorSquareF':'1stFlrSF + 2ndFlrSF',
        'TotalBsmtFinSF':'BsmtFinSF1 + BsmtFinSF2',
        'TotalBsmtBath':'BsmtFullBath + BsmtHalfBath',
        'TotalBathroom':'FullBath + HalfBath',
        'Garage_Age':'YrSold - GarageYrBlt',
        'Per_Fireplaces_Room': 'Fireplaces / TotRmsAbvGrd'}

numAtt_to_catAtt1 = {'Cat_MSSubClass':'MSSubClass'}
```

Listing 2..9. Veri hazırlama adım-1

2.3.10 Veri İşleme Ve Dönüştürme

Şimdiye kadar tanıtalan kod parçalarını sıralı bir şekilde tek seferde çalıştıracak bileşenle veri setini dönüştürme işlemi gerçekleştirilecektir. Burdaki kod parçasıyla veriler, veri setine hiç bir ekleme yapmadan dönüştürülmüştür ve veri setinden öznitelik çıkarsama yöntemiyle elde edilen verilerle kombine edilerek dönüştürülmüştür. Bu sayede dönüştürme işlemi sonucunda elde edilen her farklı veri setiyle ML algoritmalarının eğitilmesi amaçlanmıştır. Bu sayede algoritmaların yaptığı tahminler arasındaki farklar incelenerek öznitelik çıkarsamada elde edilen özniteliklerin katkılarının olduğunun ya da olmadığının anlaşılması amaçlanmıştır.

```
def Construct_Pipeline(thresh=0.0, scaler_con=False):

    pipeline_1 = Pipeline([
       ('fill_miss_values', Fill_Miss_Value(drop_con=True)),
       ('tr_Cat',TextColum_To_CategoricColum()),
       ('prop_att',Select_Proportion_Col(Label=Train_set_Labels)),
       ('scaler', Scaler(min_max=scaler_con)),
       ('column_reduction',Column_Reduction(Label=Train_set_Labels,
                              thresh=thresh, train_con=True))])


    drop_Num_List1 = ['MSSubClass']
    pipeline_2 = Pipeline([
      ('fill_miss_values', Fill_Miss_Value(drop_con=True)),
      ('feature_combine', Feature_Combine_Attribute_Adder(
                      operation_atts = numAtt_to_catAtt1)),
      ('tr_Cat',TextColum_To_CategoricColum()),
```

```
      ('prop_att',Select_Proportion_Col(Label=Train_set_Labels)),
      ('scaler', Scaler(min_max=scaler_con)),
      ('column_reduction',Column_Reduction(Label=Train_set_Labels,
                      thresh=thresh, train_con=True,
                      drop_con=True, drp_list=drop_Num_List1))])


    drop_Num_List2 = ['YearBuilt','YearRemodAdd',
                                'GarageYrBlt','YrSold']
    pipeline_3 = Pipeline([
      ('fill_miss_values', Fill_Miss_Value(drop_con=True)),
      ('feature_combine', Feature_Combine_Attribute_Adder(
                  operation_atts = arithmetic_operation1)),
      ('tr_Cat',TextColum_To_CategoricColum()),
      ('prop_att',Select_Proportion_Col(Label=Train_set_Labels)),
      ('scaler', Scaler(min_max=scaler_con)),
      ('column_reduction',Column_Reduction(Label=Train_set_Labels,
                      thresh=thresh, train_con=True,
                      drop_con=True, drp_list=drop_Num_List2))])


    return (pipeline_1, pipeline_2, pipeline_3)

def fit_transform(Construct_Pipeline=None):
    pip_dict = {}

    for i, pipe in zip((1, 2, 3), Construct_Pipeline):
        Train_set = Train_set
        pip_dict[i] = pipe.fit_transform(Train_set)

    return pip_dict
```

Listing 2..10. Pipline ile veri hazırlama adım-2

2.3.11 Dönüştürme İşlemi Sonucu

Dönüştürme işleminin başarılı bir şekilde gerçekleştiğini göstermek adına dönüştürme sonucunda elde edilen dört veri setinden sadece bir tanesi gösterilmiştir.

```
standard_fit_transform_dict[1]
```

	LotFrontage	LotArea	OverallQual	YearBuilt	YearRemodAdd
1	-0.220875	-0.207142	0.651479	1.050994	0.878668
2	0.460320	-0.091886	-0.071836	0.156734	-0.429577
3	-0.084636	0.073480	0.651479	0.984752	0.830215
4	-0.447940	-0.096897	0.651479	-1.863632	-0.720298
5	0.641972	0.375148	1.374795	0.951632	0.733308
...	...	...	...	...	...
1456	-0.357114	-0.260560	-0.071836	0.918511	0.733308
1457	0.687385	0.266407	-0.071836	0.222975	0.151865
1458	-0.175462	-0.147810	0.651479	-1.002492	1.024029
1459	-0.084636	-0.080160	-0.795151	-0.704406	0.539493
1460	0.233255	-0.058112	-0.795151	-0.207594	-0.962566

Şekil 2.2. Dönüştürülmüş veri seti

2.3.12 Eğitim ve Test Setine Ayırma

Veri setini dönüştürüldükten sonra veri setindeki verilerin hedef değişkenle(Train_Label_y) arasındaki korelasyonu incelendiğinde en yüksek korelasyona sahip **OverallQual** olduğundan veri setini train test setine ayırma işlemi bu özniteliğe göre yapılırsa tüm veri setini iyi bir şekilde temsil eden **test seti** elde edilmiş olur.(Géron, 2019)
Burdan yola çıkarak aşağıdaki kod parçasıyla bahsedilen uygulama gerçekleştirilmiştir.

```
prepared_data =  fit_transform(Construct_Pipeline=
                Construct_Pipeline(thresh=0.2,
                scaler_con=False), train_set=Train_set)

prepared_data[1]['SalePrice']=Train_set_Labels

corr_matrix = prepared_data[1].corr()

corr_matrix['SalePrice'].sort_values(ascending=False)
```

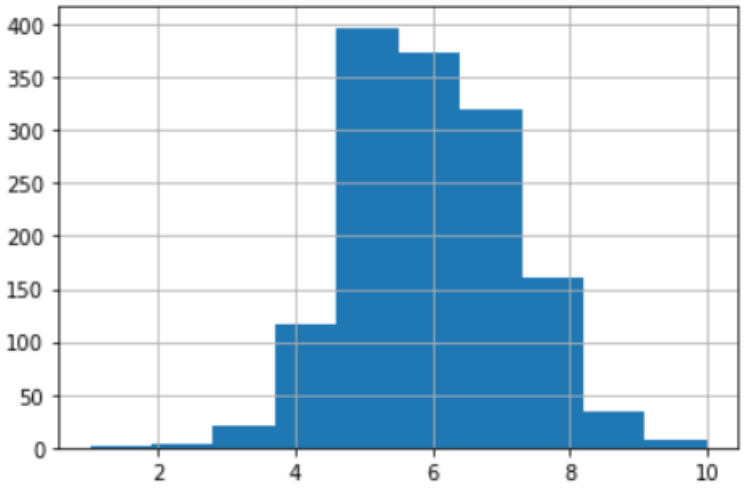
Listing 2..11. Hazırlanmış veri setinin korelasyon değerleri

```
SalePrice              1.000000
OverallQual            0.794784
Ratio_WoodFlSF_cat     0.745681
Ratio_WoodFlSF         0.696244
TotalFloorSquareF      0.674113
                          ...
AfterRemodAge         -0.540575
KitchenQual_TA        -0.544008
GarageFinish_Unf      -0.549215
House_Age             -0.558680
ExterQual_TA          -0.608058
Name: SalePrice, Length: 78, dtype: float64
```

Şekil 2.3. Verilerin, SalePrice ile arasındaki korelasyon değerleri

Train_set[’OverallQual’] histogramı incelendiğinde değerler 4.5 ile 8 değerleri arasında toplanmış olduğu rahatlıkla görülebilir, bu yüzden veri setini bu özniteliğe göre eğitim ve test setine ayırmak daha faydalı olacaktır.

Not: Aslında bu yaklaşım verilerin sürekli olduğu öznitelik üzerinden uygulanması veri setini çok daha iyi bir şekilde temsil eden test setini verecektir.



Şekil 2.4. OverallQual histogramı

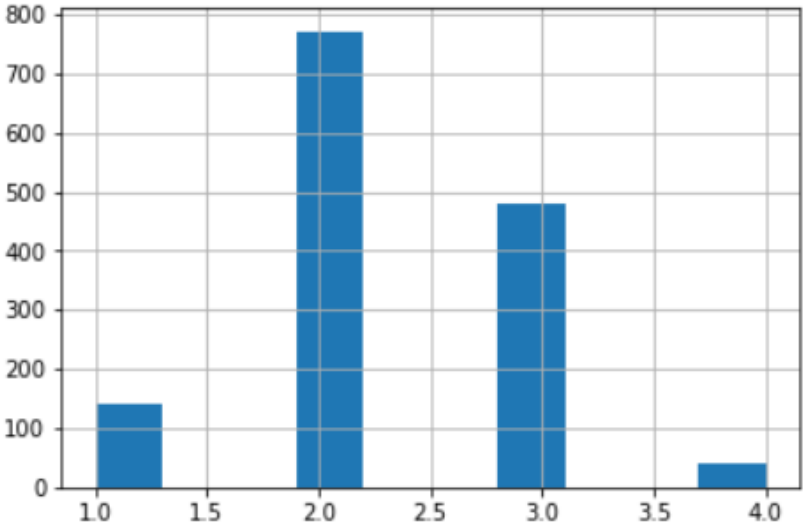
```
prepared_data[1]["OverallQual_cat"] =
                pd.cut(Train_set['OverallQual'],
                bins=[0., 4, 6., 8., np.inf],
```

```
                    labels=[1, 2, 3, 4,])

prepared_data[1]["OverallQual_cat"].fillna(
                          method='ffill', inplace=True)

prepared_data[1]["OverallQual_cat"].hist()
```

Listing 2..12. OverallQual verilerini kategorize ederek yeni sutun oluşturma



Şekil 2.5. OverallQual_cat özniteliğinin histogramı

```
prepared_data[1]["OverallQual_cat"].value_counts()
```

```
2    771
3    479
1    141
4     41
Name: Ratio_WoodFlSF_cat, dtype: int64
```

Şekil 2.6. OverallQual_cat özniteliğindeki değerlerinin adedi

Son olarak veri setini eğitim ve test setine ayırma işlemi için oluşturulan **Split_Data_To_Train_Test_Set** sınıfı kullanılacaktır. Ayırma işleminin ardından test setinin veri setini ne kadar iyi temsil ettiğini ölçmek için scikit-learn kütüphanesinin **train_test_split** bileşenin ayırdığı test seti arasında kıyaslama yapılacaktır.

```
Strat_train_set, Strat_test_set=Split_Data_To_Train_Test_Set(
```

```
        Label=Train_set["OverallQual_cat"])
        .fit_transform(prepared_data[1])

def Ratio_WoodFlSF_Cat_proportion(data):
    return data['OverallQual_cat'].value_counts() / len(data)

_, test_set = train_test_split(prepared_data[1],
                               test_size=0.2, random_state=42)

compare_props = pd.DataFrame({
    'OverAll': Ratio_WoodFlSF_Cat_proportion(prepared_data[1]),
    'Stratified': Ratio_WoodFlSF_Cat_proportion(Strat_test_set),
    'Random': Ratio_WoodFlSF_Cat_proportion(test_set)
}).sort_index()

compare_props['Random %Err'] =        100 * compare_props['Random']
                                    / compare_props['OverAll'] - 100
compare_props['Stratified %Err']=100 * compare_props['Stratified']
                                    / compare_props['OverAll'] - 100
```

Listing 2..13. Veri setini OverallQual_cat özniteliğine göre eğitim ve test setine ayırma

```
compare_props
```

	OverAll	Stratified	Random	Random %Error	Stratified %Error
1	0.098464	0.090592	0.080139	-18.610226	-7.994168
2	0.538408	0.540070	0.536585	-0.338490	0.308663
3	0.334497	0.344948	0.348432	4.165909	3.124250
4	0.028631	0.024390	0.034843	21.696269	-14.812612

Şekil 2.7. Stratified test seti ile random test seti karşılaştırılması

Şekil 2.10 incelendiğinde StratifiedShuffleSplit ile ayrılan test setinin random train_test_split ile elde edilen test setinden daha iyi bir şekilde veri setini temsil ettiği hata oranlarından rahatlıkla anlaşılabilir.

2.3.13 Makine Öğrenmesi Modellerinin Eğitilme Aşaması

Bu aşamada yukarıda elde ettiğimiz üç veri setinin her birini üç ayrı ML modelininin her biri eğitilerek r2 scorları elde edilmiştir. Bu sayede üç veri seti için ML algoritmalarının performamsı incelenecektir.

```
prepared_data[1].drop(['SalePrice'], axis=1, inplace=True)

lin_reg = LinearRegression()
```

```
tree_reg = DecisionTreeRegressor(random_state=42)
rnd_forest_reg = RandomForestRegressor(n_estimators=100,
                                       random_state=42)

def model_training(pip_dict=None, train_single_models=True,
                   train_index=None,test_index=None, label=None):

    per_pipe_scores = {}

    for i, pipe in pip_dict.items():

        per_predictor = {}

        for name, model in zip(('lin_reg','tree_reg',
                                              'rnd_forest_reg'),
                                             (lin_reg, tree_reg,
                                               rnd_forest_reg)):

            if (train_single_models):

                model.fit(pipe.loc[train_index].values,
                                          label.loc[train_index])

                predicet_y = model.predict(
                                       pipe.loc[test_index].values)

                per_predictor[name] = r2_score(
                                label.loc[test_index], predicet_y)

            else:
                scores = cross_val_score(model, pipe.values, label,
                                             scoring='r2', cv=10)
                per_predictor[name] = scores

        per_pipe_scores['Pipe_' +  str(i)] = per_predictor

    return per_pipe_scores
```

Listing 2..14. Ml model eğitimi ve testi

Son aşama olarak aşağıdaki kod parçası eğitilen modellerden elde edilen skorların daha düzgün ve anlaşılır bir şekilde incelenmesi için sonuçları veri çerçevesine ekleyecektir.

```
def monitor_result(per_pipe_scores=None,train_single_models=False):

    if (train_single_models):
        return pd.DataFrame(per_pipe_scores)

    else:
        df=pd.DataFrame()
        col1 = []
        col2 = []
```

```
        for name, val in per_pipe_scores.items():
            col1.append(name)
            for pred, scores in val.items():
                col2.append(pred)
                df = pd.concat([df,pd.DataFrame(scores)], axis=1)

        new_col1 = [item for item in col1
                                 for i in range(len(set(col2)))]

        Scores_DF=pd.DataFrame(df.values,columns=[new_col1,col2])

        Statistics_DF=pd.concat([Scores_DF.mean(axis=0),
                                 Scores_DF.std(axis=0)], axis=1)
        Statistics_DF.columns=['Mean', 'Std']

        return (Scores_DF, Statistics_DF)
```

Listing 2..15. Sonuç hazırlama ve gözlemleme

Yukarıda tüm aşamaların çoğu kod parçalarının tanımını yapmaktan ibaretti artık tüm kodu tek seferde çalıştıracak veri setini tek seferde dönüştürüp ve dönüştürülen veri setiyle Machine Learning (ML) modellerini eğitip test edip sonuçları döndüren kod parçası aşağıda listelenmiştir.

```
standard_fit_transform_dict = fit_transform(Construct_Pipeline(
                thresh=0.2, scaler_con=False), train_set=Train_set)

standard_scores = model_training(
                            pip_dict=standard_fit_transform_dict,
                            train_single_models=True,
                            train_index=train_i, test_index=test_i,
                            label=Train_set_Labels)

standard_scores_df = monitor_result(
                            per_pipe_scores=standard_scores,
                            train_single_models=True)
```

Listing 2..16. Verileri dönüştürme ve ML modellerini eğitme

2.3.14 Eğitilen ML Modellerinin Skorları ve Analizi

Eğitilen her modelin skor tablosu:

```
standard_scores_df
```

	Pipe_1	Pipe_2	Pipe_3
lin_reg	0.640831	0.640831	0.641313
tree_reg	0.687232	0.687232	0.675439
rnd_forest_reg	0.868677	0.868677	0.870259

Şekil 2.8. Eğitilen her ML modeli için elde edilen skor

Açıklama :
Öcelikle tablodaki satır ve sutun isimlerinin tanıtılması tablonun incelenmesini kolaylaştıracaktır.

- Sutun Pipe_1, Veri setine hiç bir ekleme ya da veri setinden çıkarma yapılmadan direkt dönüştürülüp ML modellerinin eğitilmesinde kullanılmıştır.
- Sutun Pipe_2, Veri setinde sadece MsSubClass sutunu kategorize edilerek ML modelleri eğitilmiştir.
- Sutun Pipe_3, Veri setine öznitelik mühendisliğiyle elde edilen yeni özniteliklerin eklenmesiyle ML modelleri eğitilmiştir. Yeni öznitelikler listesi **Listing 2..9.**'de arithmetic_operation1 sözlüğünde yer almaktadır.

Analiz : Şimdi değerlendirilebilir hale geldi; her model için analiz yapılırsa:

- Öncelikle Pipe_2 sutununa bakılırsa Pipe_1 sutunu ile tamamen aynı skorlar elde edilmiştir, hatırlatmak gerekirse MsSubClass sutunu kesikli değişken verilere sahipti ve toplam farklı değer 15 tane idi, şu soru direkt akla gelebilir: madem skora katkısı yoksa neden tabloda gösterildi, tabiki de haklı bir soru ancak unutlmamalı ki bu tezçalışmasının amacı araştırma odaklı olması. MsSubClass'ı kadegorize ettiğimizde bile skorda değişiklik yaratmaması şu

anlama gelir her ne kadar skora katkısı olmasada MsSubClass, kategorik bir öznitelik olarak kullanılabilir.

- Linear Regressor ve Random Forest Regressor modelleri ile eğitilen üç farklı veri setindeki skorlar incelendiğinde Pipe_2 sutunu görmezden gelinirse soldan sağa doğru skorlarda çok küçük de olsa artış olmuştur, bunun anlamı, öznitelik mühendisliğinde üretilen özniteliklerin az da olsa modellerin iyi tahminlerde bulunmasına katkısı olmuştur.

3. SONUÇ VE DEĞERLENDİRME

Bu tez projesinde veri analizi teknikleri kullanılarak büyük verinin analiz edilerek veri setinin makine öğrenmesi algoritmalarının eğitilmesinde kullanılması ve eğitilen algoritmalarının preformansları ölçülmüştür. Proje öncesinde benzer projeler araştırılmış, makaleler ve farklı yaklaşımlar incelenmiştir. Yapılan literatür taramasında sayısız çalışma incelenmiştir ve tüm çalışmalarda veri analizi süreci ve makine öğrenmesi modellerinin eğitilme süreci aynı aşamalardan oluştuğu gözlemlenmiştir.

Proje için geliştirilen uygulamada veri seti, veri analizi sonucunda dönüştürülen veri seti, ML algoritmalarının eğitilmesinde kullanılmıştır, kullanılan ML algoritmaları: Lineer Regression, Decision Tree ve Random Forest Regressor. Uygulama geliştirme ortamı olarak Jupyter Notebook kullanılmıştır. Python ve veri analizi için gerekli kütüphaneler(Numpy, Pandas, Matplotlib ve Scikit-learn) bu projenin kalbini oluşturmaktadır. Yapılan geliştirmeler sonucunda hazırlanan veri seti üç farklı makine öğrenmesi algoritmasının eğitilmesinde kulanılmıştır ve performansları ölçülmüştür. Öznitelik mühendisliği ile üretilen özniteliklerin veri setiyle kombine edildiğinde çok küçük de olsa ML modellerinin yaptığı tahminlerde artış sağladığı gözlenmiştir.

Geliştirilen bu kapsamlı projede rastlanan olumsuz durumlar bulunmaktadır. ML algoritmaları çok küçük veri setleriyle eğitildikten sonra tahminleme yaptığında gerçek değerle tahmin edilen değer arasında büyük farklar olabiliyor, bunun anlamı ML algoritması yeteri kadar veriyle eğitilmediğinden algoritma iyi öğrenmemiştir(underfitting) demektir. Sonuç olarak veri seti çok küçük ise daha güçlü ML modeli seçilmeli, **Şekil 2.11.** random_forest modelinin performansı diğer iki modelden çok daha iyi olduğu rahatlıkla görülebilir bu da küçük veri seti için daha güçlü model tercih edilmeli sonucuna ulaştırır bizi.

KAYNAKLAR DİZİNİ

Ali, P.J.M. and Faraj, R.H., 2014, Metodolojiye Yaklaşımlar, *Data Normalization and Standardization: A Technical Report*, p. 1–6.

Bakshi, C., 2020, Random Forest Regressor, URL `https://levelup.gitconnected.com/random-forest-regression-209c0f354c84`, (Erişim tarihi:26 Ocak 2022).

Durna, M.B., 2020a, Cross Validation, *Cross-Validation nedir? Nasıl çalışır?*

Durna, M.B., 2020b, GridSearchCV, *Hiperparametre Optimizasyonu*.

Grus, J., 2019, Decision Tree Regressor, *Data Science from Scratch*.

Géron, A., 2019, Eğitim ve Test Setine Ayırma, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow SECOND EDITION*.

Hunter, J.D., 2003, Matplotlib, URL `https://tr.wikipedia.org/wiki/Pandas#:~:text=pandas%2C%20veri%20i%C5%9Flemesi%20ve%20analizi,veri%20yap%C4%B1s%C4%B1%20%C3%BCzerinde%20ger%C3%A7ekle%C5%9Ftirilebilir%20olur.`

Mamur, H., AKTAŞ, A. and KUZEY, S., 2017, Metodolojiye Yaklaşımlar, *Learning Feature Engineering for Classification*, p. 2529–2535.

McKinney, W., 2008, Pandas, URL `https://en.wikipedia.org/wiki/Scikit-learn`, (Son güncelleme tarih: 17 Mart 2020),(Erişim Tarihi:25 Ocak 2022).

Oliphant, T., 2016, Numpy, URL `https://en.wikipedia.org/wiki/Scikit-learn`, (Son güncelleme tarih: 10 Mayıs 2021),(Erişim Tarihi:25 Ocak 2022).

van Rossum, G., 1991, Python Programlama Dili, URL `https://en.wikipedia.org/wiki/Python_(programming_language)`, (Son güncelleme tarih: 14 Ocak 2022),(Erişim Tarihi:25 Ocak 2022).

Vikipedi, 2007, Scikit-learn, URL `https://en.wikipedia.org/wiki/Scikit-learn`, (Son güncelleme tarih: 25 Ekim 2021),(Erişim Tarihi:25 Ocak 2022).

Şevket Ay, 2020, R2 Score, *Model Performansını Değerlendirmek — Metrikler*.