


KNIME İLE İKİNCİ EL OTO VERİSİ REGRESYON ANALİZİ



Mert AÇIKGÖZ
H. Fatih AKTAŞ
İsa ERGÜN
Mesut HARDAL
Emre MENOKAN
Oğuzhan TANRIKULU

Tanımlayıcı İş Analitiği Yöntemleri – İş Analitiği YL.
İstanbul Şehir Üniversitesi - 2019

Projenin Amacı

Projenin amacı www.kaggle.com dan alınan ikinci el oto verisinin Knime programını kullanarak araçların fiyatlarının tahmininin yapılmasıdır. Tahminler regresyon yöntemleri kullanılarak yapılmıştır. Proje tüm dünyada kabul görmüş CRISP-DM metodolojisi kullanılarak gerçekleştirilmiştir. Proje kapsamında hazırlanan workflow, Knime analitik platformunda yapılmıştır. Bununla beraber Knime programına entegre edilen extension ile Python programlama dilinden ve Python'ın Pandas, numpy, matplotlib, seaborn, math ve scikit-learn gibi hazır kütüphanelerinden faydalanılmıştır.

CRISP-DM Metodolojisi

Veri madenciliği birçok farklı aşamadan oluşur. Genellikle tüm veri madenciliği çalışmalarında verilerin kaynaklardan toplanması ve entegrasyonu, verilerin temizlenmesi, modelin oluşturulması, modelin denenmesi ve sonuçların sunuma hazırlanması adımları karşımıza çıkar.

CRISP-DM (Cross Industry Standard Process Model for Data Mining) endüstri ve kullanılan yazılımdan bağımsız bir veri madenciliği süreç modelidir. CRISP-DM veri madenciliği süreçlerini, kullanılan yazılımdan ve endüstriden bağımsız standartlaştırmayı amaçlar.

CRISP-DM aşağıdaki fazlardan oluşur:

- **İşi anlama**

Bu adımda projenin hedeflerini ve ihtiyaçlarını iş perspektifinden anlamak, bu bilgiyi veri madenciliği sürecinde problem tanımı ve iş ön planı oluşturmak için kullanmak yer alır. Hedeflere şirket açısından bakılıp varılmak istenen noktayı anlamak ve bu problemi veri madenciliği problemine indirgemek önemlidir. Bunlar yapılırken sahip olunan verileri, yazılımları, donanımları ve proje sırasında ortaya çıkabilecek olası problemleri belirlemek gerekmektedir.

- **Veriyi Anlama**

Veri toplamak, veriyi tanımak, veri kalitesini belirleyip sorunlarını tanımlamak, gizli bilgi için hipotez oluşturmak gibi adımlardan oluşur.

- **Veriyi Hazırlama**

Verinin işlenmesi, eksik verilerin tahmini, verinin yapılandırılması, entegrasyonu, modellemelere uygun şekillendirilmesi adımlarından oluşur. Başlangıç veri kümesinden itibaren son veri kümesini oluşturmak için yapılan tüm aşamaları barındırır.

- **Modelleme**

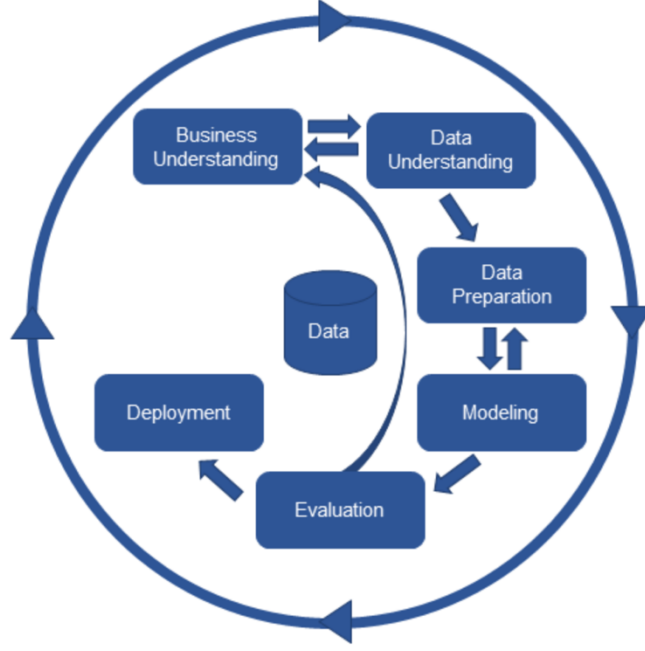
Modelleme tekniği seçilir ve yazılıma bağlı olarak hangi algoritmanın kullanılacağı da belirlenir. Nöral ağlar gibi bazı teknikler veri yapısına ilişkin özel gereksinimlere sahip olduğundan veri hazırlama aşaması burada tekrar döngüye girebilir. Modellerin parametrelerinin ve testlerinin belirlenmesi bu aşamada yer alır.

- **Değerlendirme**

Sonuçların test değerlerine göre teknik açıdan, proje aşamalarına göre de iş açısından iki farklı yönden değerlendirilmesini kapsar. Kayıp fonksiyonlara bağlı olarak bir veya daha fazla model oluşturulduğunda bunların yüksek kalite açısından değerlendirilebilmesi gerekir. Bunun için test sonuçlarına bakılarak en verimli modelin seçilmesi bu aşamaya girmektedir.

- **Dağıtım, Konumlandırma**

İş sorunlarının çözümü amacıyla modelin bir kod temsilini, işletim sistemine dağıtma aşamasıdır. En önemlisi bu kod temsili, modellemeye giden tüm veri adımlarını da içermelidir. Böylece model ham verileri model geliştirme sırasında olduğu gibi ele alacaktır. Sistemin sağlıklı çalışması için bakım planlaması ve nihai raporun sunulması ile proje sonlandırılır.



1. İşi Anlama

Kaggle'dan alınan CraigslistVehicles verisinde araçların fiyatları, silindir sayıları, motor tipleri, yakıt sistemleri, renkleri gibi çeşitli kolonlar mevcuttur. Bu kolonların detaylarından "Data Understanding" bölümünde bahsedilecektir. Bu kolonlardan fiyat kolonu bizim bağımlı değişkenimiz, diğer kolonlar da bağımsız değişkenlerimizdir. Eğer amacımızı detaylandırmamız gerekirse, bağımsız değişkenleri kullanılarak bağımlı değişkenimizi tahmin etmeye yarayacak regresyon modelleri kurmak ve bu modelleri karşılaştırmaktır.

Craigslist, dünyanın en büyük ikinci el satılık araç koleksiyonudur, Kaggle'da kullanılan bu veri, ABD'de Craigslist'te kullanılan her araç girişini içerecek şekilde genişletilmiştir.

2. Veriyi Anlama

Öncelikle “.csv” formatında bulunan veri Knime’in “Csv Reader” nodu ile okutuldu. Veri düzgün bir biçimde okunamadığı için farklı okuma nodları da denendi. “Excel Reader” harici veri okutma nodlarında başarı sağlanamadı. Veri Excel’in max row limitine takıldığı için aşağıdaki kod ile 9 parçaya bölündü.

```
def chunks(l, n):
    """Yield successive n-sized chunks from l."""
    for i in range(0, len(l), n):
        yield l[i:i + n]
```

```
import numpy as np

def split(df, chunk_size):
    indices = index_marks(df.shape[0], chunk_size)
    return np.split(df, indices)

chunks = split(df, 65520)
for c in chunks:
    print("Shape: {}; {}".format(c.shape, c.index))
```

```
Shape: (65520, 22); RangeIndex(start=0, stop=65520, step=1)
Shape: (65520, 22); RangeIndex(start=65520, stop=131040, step=1)
Shape: (65520, 22); RangeIndex(start=131040, stop=196560, step=1)
Shape: (65520, 22); RangeIndex(start=196560, stop=262080, step=1)
Shape: (65520, 22); RangeIndex(start=262080, stop=327600, step=1)
Shape: (65520, 22); RangeIndex(start=327600, stop=393120, step=1)
Shape: (65520, 22); RangeIndex(start=393120, stop=458640, step=1)
Shape: (65520, 22); RangeIndex(start=458640, stop=524160, step=1)
Shape: (1679, 22); RangeIndex(start=524160, stop=525839, step=1)
```

Ardından veriler aşağıdaki kod ile ayrı ayrı excel dosyalarına .xlsx formatında yazdırıldı.

```
df.iloc[393120:458640].to_excel(r'/Users/mesuthardal/Documents/craigslistVehiclespart7.xlsx')
```

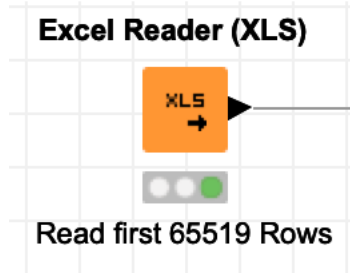
Verinin baştaki satır ve sütun sayısı ve sütun isimleri aşağıdaki gibidir.

```
#satir ve sutun gosterimi
df.shape

(525839, 22)
```

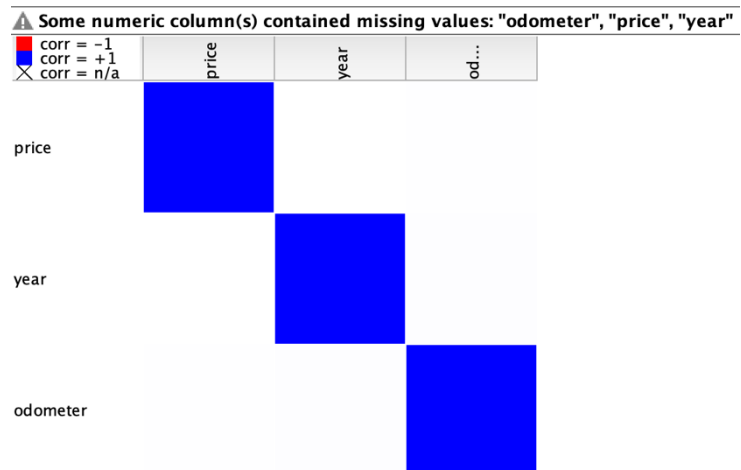
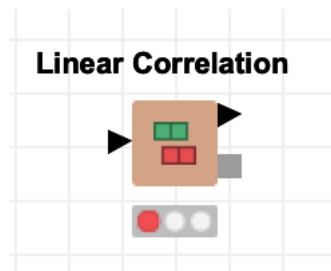
```
#kolonların gösterilmesi
df.columns

Index(['url', 'city', 'city_url', 'price', 'year', 'manufacturer', 'make',
       'condition', 'cylinders', 'fuel', 'odometer', 'title_status',
       'transmission', 'VIN', 'drive', 'size', 'type', 'paint_color',
       'image_url', 'desc', 'lat', 'long'],
      dtype='object')
```



“Excel Reader” nodları ile toplam 525.839 satır ve 22 sütün verinin tamamı okutuldu.

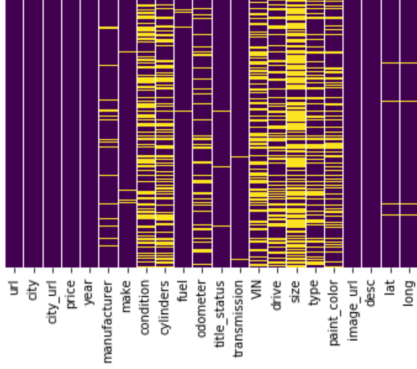
Nümerik verilerin aralarındaki korelasyonu “Linear Correlation” nodu ile görüntülendi ve aşağıdaki tablo elde edildi. Veri henüz ön işlemlere tabi tutulmadığı, tip dönüşümleri yapılmadığı ve eksik değerleri olduğu için herhangi bir korelasyon bulunamadı.



Ne kadar eksik değerimiz olduğu aşağıdaki kod ile python da grafiksel olarak görüntülendi.

```
#eksik degerlerin grafiksel gosterimi
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis') # missing value check
```

<matplotlib.axes._subplots.AxesSubplot at 0x23ca0e61eb8>



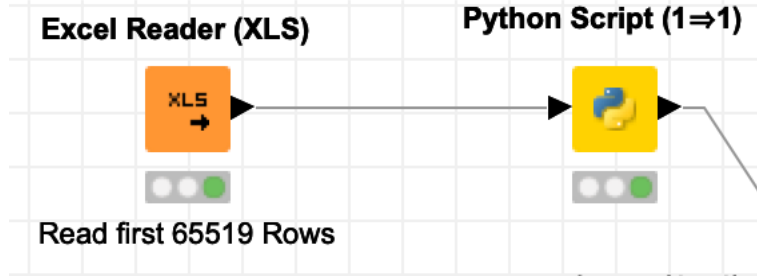
```
#kolonlardaki null degerlerin sayisal olarak gosterilmesi
for i in range(len(df.columns)):
    x = df.columns[i]
    print(x, '-', df[df[x].isnull() == True].shape)
```

```
url - (0, 22)
city - (0, 22)
city_url - (0, 22)
price - (0, 22)
year - (1487, 22)
manufacturer - (26915, 22)
make - (9677, 22)
condition - (250074, 22)
cylinders - (218997, 22)
fuel - (4741, 22)
odometer - (110800, 22)
title_status - (4024, 22)
transmission - (4055, 22)
VIN - (239238, 22)
drive - (165838, 22)
size - (366256, 22)
type - (159179, 22)
paint_color - (180021, 22)
image_url - (26, 22)
desc - (30, 22)
lat - (11790, 22)
long - (11790, 22)
```

3. Veriyi Hazırlama

Kullanılan CRISP-DM metodolojisinin üçüncü adımına göre veri tip dönüşümleri, eksik değerlerin tamamlanması, encoding işlemleri, hangi kolonların modele dahil edilip edilmeyeceğinin belirlenmesi ve outlier değerlerin temizlenmesi gibi bazı ön işleme adımlarının uygulanması gerekmektedir.

Bu aşamada extensions olarak indirilen “Python Script (1=>1)” noduna ayrı ayrı “Excel Reader” lardan gelen veriler aktarıldı. Aşağıdaki işlemler yapılarak data kullanılabilir hale getirilmeye çalışıldı.



```
#belirli kolonların silinmesi
df.drop(columns = ['url', 'city_url', 'image_url', 'size', 'VIN', 'lat', 'long'], axis=1,inplace=True)

df.shape

(525839, 15)
```

‘Manufacturer’ değerleri eksik olan satırların ‘make’ kolonundan elde edilerek doldurulması:

```
df_make_manufacturer_dropna = df.dropna(subset = ["make", "manufacturer"])
#dictionary oluşturmada önce missing value'ları drop ediyoruz, dictionary'nin hatalı oluşması için
dict_make_manufacturer = pd.Series(df_make_manufacturer_dropna.manufacturer.values,index=df_make_manufacturer_dropna.m
#make ve manufacturer için dictionary oluşturuyoruz.
df.manufacturer = df.manufacturer.fillna(df.make.map(dict_make_manufacturer))
#dictionary ile dataframe'in make alanlarını map edip, dictionary'nin manufacturer alanı ile df'in manufacturer = nan c
#güncelliyoruz
df[df.manufacturer.isnull() == True].shape #26915 - 25585 = 1330 tane missing value doldurduk

(25585, 15)
```

‘Drive’ değerleri eksik olan satırların ‘make’ kolonundan elde edilerek doldurulması:

```
df_make_drive_dropna = df.dropna(subset = ["make", "drive"])
#dictionary oluşturmada önce missing value'ları drop ediyoruz, dictionary'nin hatalı oluşması için
dict_make_drive = pd.Series(df_make_drive_dropna.type.values,index=df_make_drive_dropna.make).to_dict()
#make ve drive için dictionary oluşturuyoruz.
df.drive = df.drive.fillna(df.make.map(dict_make_drive))
#dictionary ile dataframe'in make alanlarını map edip, dictionary'nin drive alanı ile df'in drive = nan olanları
#güncelliyoruz
df[df.drive.isnull() == True].shape # 165818 - 49519 = 116299 tane missing value doldurduk

(49519, 15)
```


‘type’ değerleri eksik olan satırların ‘make’ kolonundan elde edilerek doldurulması:

```
df_make_type_dropna = df.dropna(subset = ["make", "type"])
#dictionary olusturmadan önce missing value'ları drop ediyoruz, dictionary'nin hatalı olusması için
dict_make_type = pd.Series(df_make_type_dropna.type.values,index=df_make_type_dropna.make).to_dict()
#make ve type için dictionary oluştuyoruz.
df.type = df.type.fillna(df.make.map(dict_make_type))
#dictionary ile dataframe'in make alanlarını map edip, dictionary'nin type alanı ile df'in type = nan olanları
#güncelliyoruz
df[df.type.isnull() == True].shape # 159179 - 21821 = 137358 tane missing value doldurduk

(21821, 15)
```

Odometer(kilometre) değerleri eksik olan satırlardaki araçların model yıllarına ait ortalama kilometre değerleri hesaplanarak veriler doldurulması:

```
#global olarak kullanıyoruz
dict_odometer_mean = {} # year ve mean dictionary değişkeni

# nan odometer değerleri için mean değerlerini hesaplayan fonksiyon
def calc_nanvalues_mean_func(x,y):
    global dict_odometer_mean
    if (math.isnan(y) == True):
        if x not in dict_odometer_mean:
            val = df[(df['year'] == x)]['odometer'].mean()
            return dict_odometer_mean.update({x:val})
        else:
            return dict_odometer_mean.get(x)
    else:
        return y
```

```
import math
```

```
# calc_nanvalues_mean_func fonksiyonunun her bir satır için çalıştırılması
df['odometer'] = df.apply(lambda x: calc_nanvalues_mean_func(x['year'], x['odometer']), axis=1)
```

‘price’ kolonundaki outlierların box plot quantile değerleri baz alınarak silinmesi:

```
y = df['price']
removed_outliers = y.between(y.quantile(.10), y.quantile(.90))
```

```
df = df[removed_outliers]
```

```
df.shape
```

```
(420708, 22)
```

```
#1989 öncesi ve == 2020 için veri az olduğu için alınmıyor..
df = df[(df['year'] > 1989) & (df['year'] < 2020)] #??? kayıt alındı.
```

```
df.shape
```

```
(406979, 22)
```

Regresyona verilecek kolonlardaki değerlerin anlamlı ve numerik hale getirilebilmesi için bu kolonlardaki değerlere “one hot encoding” işleminin uygulanması :

```
drive = pd.get_dummies(df['drive'], prefix = 'drive')
df.drop('drive', axis=1,inplace=True)
df = pd.concat([df, drive],axis=1)
```

```
type = pd.get_dummies(df['type'], prefix = 'type')
df.drop('type', axis=1,inplace=True)
df = pd.concat([df, type],axis=1)
```

```
transmission = pd.get_dummies(df['transmission'], prefix = 'transmission')
df.drop('transmission', axis=1,inplace=True)
df = pd.concat([df, transmission],axis=1)
```

```
paint_color = pd.get_dummies(df['paint_color'], prefix = 'paint_color')
df.drop('paint_color', axis=1,inplace=True)
df = pd.concat([df, paint_color],axis=1)
```

```
title_status = pd.get_dummies(df['title_status'], prefix = 'title_status')
df.drop('title_status', axis=1,inplace=True)
df = pd.concat([df, title_status],axis=1)
```

```
manufacturer = pd.get_dummies(df['manufacturer'], prefix = 'manufacturer')
df.drop('manufacturer', axis=1,inplace=True)
df = pd.concat([df, manufacturer],axis=1)
```

```
cylinders = pd.get_dummies(df['cylinders'], prefix = 'cylinders')
df.drop('cylinders', axis=1,inplace=True)
df = pd.concat([df, cylinders],axis=1)
```

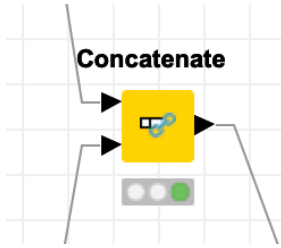
```
condition = pd.get_dummies(df['condition'], prefix = 'condition')
df.drop('condition', axis=1,inplace=True)
df = pd.concat([df, condition],axis=1)
```

```
fuel = pd.get_dummies(df['fuel'], prefix = 'fuel')
df.drop('fuel', axis=1,inplace=True)
df = pd.concat([df, fuel],axis=1)
```

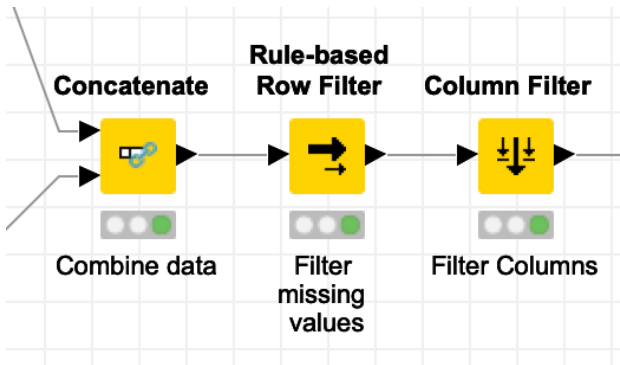
Imputation için kullanıldıktan sonra make ve desc kolonlarının hariç bırakılması:

```
df.drop('make', axis=1,inplace=True)
df.drop('desc', axis=1,inplace=True)
```

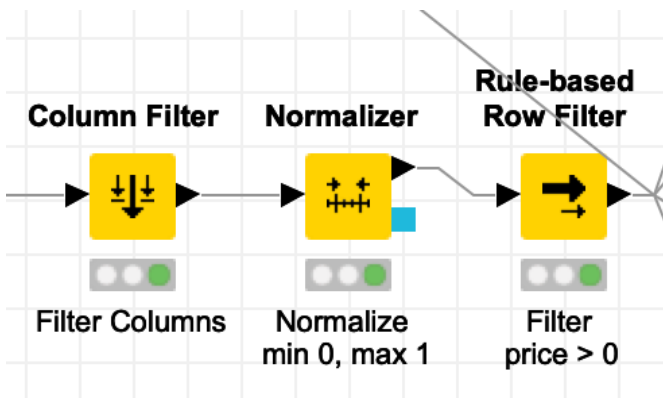
“Python Script”lerden gelen veriler “Concatenate” nodu ile birleştirildi.



Kolonlarda doldurulabilecek eksik verilerin diğer kolonlardan bulunarak tamamlanmalarının sağlanmasından sonra herhangi bir eksik veri doldurma işlemi yapılamadığından, kalan eksik verili satırlar silindi. Bu aşamada “Rule-based Row Filter” nodu kullanıldı. Buna ek olarak modellerin düzgün çalışmasını sağlamak için fazla eksik değere sahip önem teşkil etmeyecek kolonlar da “Column Filter” kullanılarak kaldırıldı.

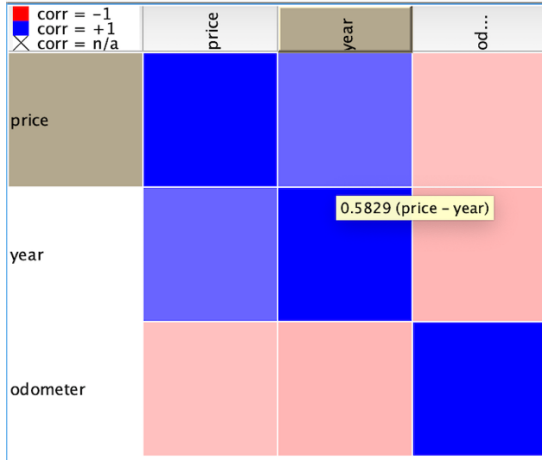


Modellerinin verimli çalışabilmesi için sonuçlar numerik değerler “Normalizer” nodu ile max 1 ve min 0 olacak şekilde normalize edildi. Ardından oluşan değerlerden bağımlı değişkenimiz “price” kolonundaki hatalı sonuç almaya neden olacağı düşünülen 0 değerleri Rule-Based Row Filter ile kaldırıldı.



Tüm bu işlemlerden sonra geriye 112.510 satır kaldı.

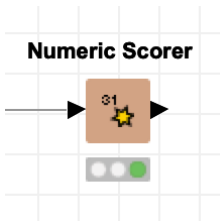
Veri ön işleme sonrası lineer korelasyon haritası:



4. Modelleme

İkinci el araç verisinin fiyat tahminlemesi için Linear Regression, Random Forest Regression, Gradient Boosted Trees Regression ve Artificial Neural Networks modelleri kullanıldı. Bunlardan en iyi R^2 sonucunu Random Forest Regression modeli verdi.

5. Değerlendirme



Modellerden sonra “Numeric Scorer” nodu ile aşağıdaki çıktılar elde edildi.

- **Linear Regression**

Numeric Scorer:

R ² :	0.646
Mean absolute error:	0.102
Mean squared error:	0.019
Root mean squared error:	0.138
Mean signed difference:	-0.006
Mean absolute percentage error:	1.29

- **Random Forest Regression**

R ² :	0.826
Mean absolute error:	0.067
Mean squared error:	0.01
Root mean squared error:	0.098
Mean signed difference:	0
Mean absolute percentage error:	1.574

- **Gradient Boosted Trees Regression**

R ² :	0.779
Mean absolute error:	0.075
Mean squared error:	0.012
Root mean squared error:	0.11
Mean signed difference:	-0.002
Mean absolute percentage error:	1.637

- **Artificial Neural Networks**

R ² :	0.788
Mean absolute error:	0.073
Mean squared error:	0.011
Root mean squared error:	0.107
Mean signed difference:	-0
Mean absolute percentage error:	1.085

- **Modellerden elde edilen sonuçların değerlendirilmesi:**

Kullanılan dört modellerden elde edilen R², mean absolute error, mean squared error ve mean absolute percentage error gibi parametreler karşılaştırıldığında başarı sıralaması aşağıdaki gibi oluşmuştur.

- 1.Random Forest Regression
- 2.Artificial Neural Networks
- 3.Gradient Boosted Trees Regression
- 4.Linear Regression

WORKFLOW:

