Öğrencinin;

ADI: MERVE

SOYADI: BALCI

NO: 1821221022

BÖLÜM: BİLGİSAYAR MÜH.

Dersin;

ADI: Introdiction to Data Science

EĞİTMEN: Dr. Gönül ULUDAĞ

Ar. Gör. Zeki Kuş

İçindekiler

Dataset definition:	Kilometre, yakıt türü gibi özellikleri kullanarak bir aracın fiyatını tahmin etmemizi sağlar. Bu veri setinde araçlarla ilgili birçok özellik bulunmaktadır, buradaki özellikleri kullanarak araç fiyat tahmini yapabiliriz.
Dataset source (web address):	https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho?ref=hackernoon.com&select=Car+details+v3.csv
Aim of the project:	özelliklerine göre araçların fiyat tahmini yapmak.

1- Project Proparsal

Bu veri setinde araçlarla ilgili birçok özellik bulunmaktadır, buradaki özellikleri kullanarak araç fiyat tahmini yapabiliriz. Bugün milyonlarca insan araç almak istiyor ve arıyor, bu tahmini yapmak herkesin işine yarayacaktır.

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8123	Hyundai i20 Magna	2013	320000	110000	Petrol	Individual	Manual	First Owner	18.5 kmpl	1197 CC	82.85 bhp	113.7Nm@ 4000rpm	5.0
8124	Hyundai Verna CRDi SX	2007	135000	119000	Diesel	Individual	Manual	Fourth & Above Owner	16.8 kmpl	1493 CC	110 bhp	24@ 1,900-2,750(kgm@ rpm)	5.0
8125	Maruti Swift Dzire ZDi	2009	382000	120000	Diesel	Individual	Manual	First Owner	19.3 kmpl	1248 CC	73.9 bhp	190Nm@ 2000rpm	5.0
8126	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5.0
8127	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5.0

8128 rows × 13 columns

Üzerinde çalıştığım datasetim 8128satır x 13 sütundan oluşmaktadır
Vehicle data setini daha iyi anlayabilmek için EDA adımlarının uygulayarak, veri setiyle alakalı daha doğru ilerleyebiliriz.

Bazı EDA adımları;

```
vehicleDf.columns # data set's columns
```

```
Index(['name', 'year', 'selling_price', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner', 'mileage', 'engine', 'max_power', 'torque',
       'seats'],
      dtype='object')
```

```
vehicleDf.nunique(axis=0) # number of unique values for each variable (index/columns)
```

```
name             2058
year               29
selling_price     677
km_driven         921
fuel                4
seller_type         3
transmission        2
owner               5
mileage           393
engine            121
max_power         322
torque            441
seats               9
dtype: int64
```

```
vehicleDf.info() # summary of a DataFrame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   name           8128 non-null   object 
 1   year           8128 non-null   int64  
 2   selling_price  8128 non-null   int64  
 3   km_driven      8128 non-null   int64  
 4   fuel           8128 non-null   object 
 5   seller_type    8128 non-null   object 
 6   transmission   8128 non-null   object 
 7   owner          8128 non-null   object 
 8   mileage        7907 non-null   object 
 9   engine         7907 non-null   object 
 10  max_power      7913 non-null   object 
 11  torque         7906 non-null   object 
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
```

bazı kolonlarda yanlarında yazan birimlerden dolayı float değil object olarak gözüküyor, birimleri temizleyerek, tür değişimi yapılmalı;

```
vehicleDf['mileage']= vehicleDf['mileage'].str.replace('kmpl','')
```

```
vehicleDf['mileage']= vehicleDf['mileage'].str.replace('km/kg','')
vehicleDf['mileage']=vehicleDf['mileage'].astype(float)
```

```
vehicleDf['engine']= vehicleDf['engine'].str.replace('CC','').astype(float)
```

```
vehicleDf['max_power']=vehicleDf['max_power'].str.replace('bhp','')
```

```
vehicleDf[['max_power']] = vehicleDf[['max_power']].replace('(-?[^\d\.])', '', regex=True).replace('', float('NaN')).astype(float
```

```
vehicleDf.info() # summary of a DataFrame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           8128 non-null   object
 1   year           8128 non-null   int64
 2   selling_price  8128 non-null   int64
 3   km_driven      8128 non-null   int64
 4   fuel           8128 non-null   object
 5   seller_type    8128 non-null   object
 6   transmission   8128 non-null   object
 7   owner          8128 non-null   object
 8   mileage        7907 non-null   float64
 9   engine         7907 non-null   float64
 10  max_power      7912 non-null   float64
 11  torque         7906 non-null   object
 12  seats          7907 non-null   float64
dtypes: float64(4), int64(3), object(6)
memory usage: 825.6+ KB
```

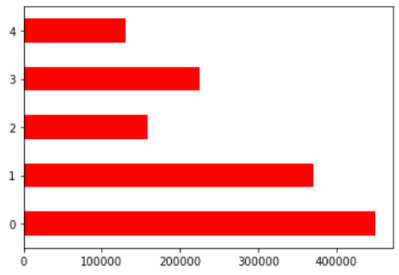
Describe() ile kolonların ortalama, standart sapma vs. özellikleri

```
vehicleDf.describe()  # descriptive statistics.
```

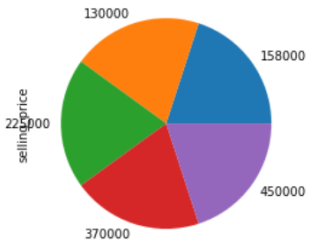
	year	selling_price	km_driven	mileage	engine	max_power	seats
count	8128.000000	8.128000e+03	8.128000e+03	7907.000000	7907.000000	7912.000000	7907.000000
mean	2013.804011	6.382718e+05	6.981951e+04	19.418783	1458.625016	91.517919	5.416719
std	4.044249	8.062534e+05	5.655055e+04	4.037145	503.916303	35.822499	0.959588
min	1983.000000	2.999900e+04	1.000000e+00	0.000000	624.000000	0.000000	2.000000
25%	2011.000000	2.549990e+05	3.500000e+04	16.780000	1197.000000	68.050000	5.000000
50%	2015.000000	4.500000e+05	6.000000e+04	19.300000	1248.000000	82.000000	5.000000
75%	2017.000000	6.750000e+05	9.800000e+04	22.320000	1582.000000	102.000000	5.000000
max	2020.000000	1.000000e+07	2.360457e+06	42.000000	3604.000000	400.000000	14.000000

“selling_price” kolonunun ilk 5 değerlerinin grafiksel olarak gösterimi

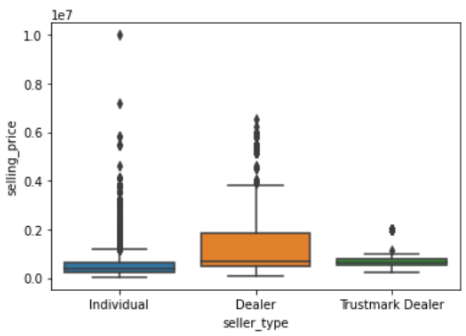
```
<AxesSubplot:>
```

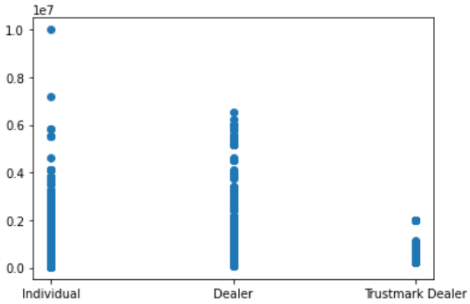



```
<AxesSubplot:ylabel='selling_price'>
```



“seller_type” kolonuna göre “selling_price” boxplot ve scatter plot gösterimi





Veri setimi okuyup, incedikten sonra elimdeki araç özellikleri kolonlarını kullnarak bir tahmin yapacağım. Kullanacağım makine öğrenimi modeli regresyon çünkü hedefim tahmin ve öngörü. Regresyon, bağımlı bir değişken ile bir veya daha fazla bağımsız değişken arasındaki ilişkiyi tahmin etmek için kullanılan bir dizi istatistiksel yaklaşımdır, yapacağım şey araç özelliklerini kullanarak fiyatı tahmin etmek .

2- Exploratory Data Analysis

Veri setimdeki kolonlarımın türlerini incelemiştim şimdi nümreik ve kategorik olarak ayırdım

Veri setimdeki nümerik özeliğe sahip kolonlarım

	year	selling_price	km_driven	mileage	engine	max_power	seats
0	2014	450000	145500	23.40	1248.0	74.00	5.0
1	2014	370000	120000	21.14	1498.0	103.52	5.0
2	2006	158000	140000	17.70	1497.0	78.00	5.0
3	2010	225000	127000	23.00	1396.0	90.00	5.0
4	2007	130000	120000	16.10	1298.0	88.20	5.0
...	...	...	...	...	...	...	...
8123	2013	320000	110000	18.50	1197.0	82.85	5.0
8124	2007	135000	119000	16.80	1493.0	110.00	5.0
8125	2009	382000	120000	19.30	1248.0	73.90	5.0
8126	2013	290000	25000	23.57	1396.0	70.00	5.0
8127	2013	290000	25000	23.57	1396.0	70.00	5.0

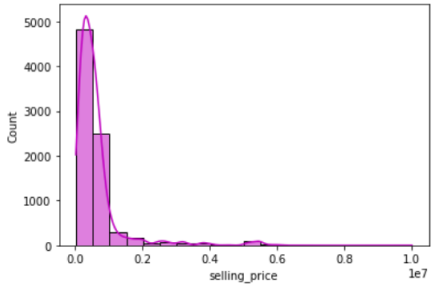
8128 rows × 7 columns

Nümerik olmayan kolonlarım;

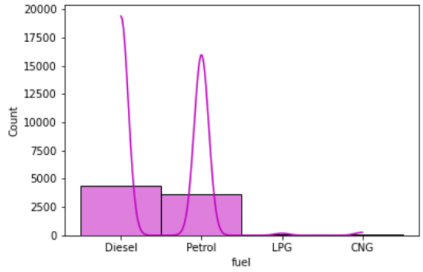
	name	fuel	seller_type	transmission	owner	torque
0	Maruti Swift Dzire VDI	Diesel	Individual	Manual	First Owner	190Nm@ 2000rpm
1	Skoda Rapid 1.5 TDI Ambition	Diesel	Individual	Manual	Second Owner	250Nm@ 1500-2500rpm
2	Honda City 2017-2020 EXi	Petrol	Individual	Manual	Third Owner	12.7@ 2,700(kgm@ rpm)
3	Hyundai i20 Sportz Diesel	Diesel	Individual	Manual	First Owner	22.4 kgm at 1750-2750rpm
4	Maruti Swift VXI BSIII	Petrol	Individual	Manual	First Owner	11.5@ 4,500(kgm@ rpm)
...	...	...	...	...	...	...
8123	Hyundai i20 Magna	Petrol	Individual	Manual	First Owner	113.7Nm@ 4000rpm
8124	Hyundai Verna CRDi SX	Diesel	Individual	Manual	Fourth & Above Owner	24@ 1,900-2,750(kgm@ rpm)
8125	Maruti Swift Dzire ZDi	Diesel	Individual	Manual	First Owner	190Nm@ 2000rpm
8126	Tata Indigo CR4	Diesel	Individual	Manual	First Owner	140Nm@ 1800-3000rpm
8127	Tata Indigo CR4	Diesel	Individual	Manual	First Owner	140Nm@ 1800-3000rpm

8128 rows × 6 columns

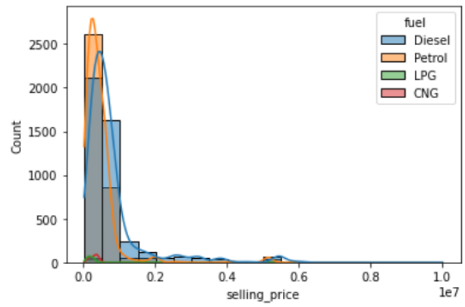
“selling_price” kolonum



“fuel” kolonum

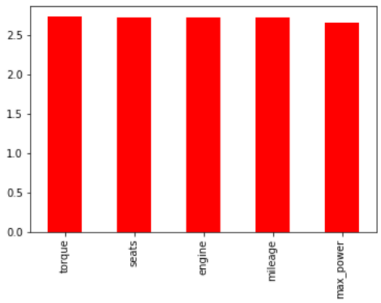


Nümerik olan “selling_price” özelliğim ve kategorik özelliğim olan “fuel” özelliğim arasındaki ilişki;



3- Preprocessing

Regresyon işlemlerini gerçekleştirmeden önce veri setimdeki eksik “Null” olan değerleri tespit edip, buna göre eksik değerler için uygun bir çözüm uygulanmalı



Eksik değerlere sahip olan kolonlarım, bu kolonlardan nümerik olanları, her bir kolondaki eksik değerleri o kolonun ortalamasıyla doldurmak iyi bir çözüm olacaktır.

```
numeric.isna().sum() #in my num
```

```
year              0
selling_price     0
km_driven         0
mileage         221
engine          221
max_power       216
seats           221
dtype: int64
```

nümerik kolonlardaki toplam eksik değerlerin sayısı

```
numeric.mean() # i can fill it with columm
```

```
year               2013.804011
selling_price    638271.807702
km_driven         69819.510827
mileage              19.418783
engine             1458.625016
max_power            91.517919
seats                 5.416719
dtype: float64
```

kolonların ortalama hesapları

```
vehicleDf = vehicleDf.fillna(numeric.mean()) #with fillna(), i fill my columns, columns's mean
vehicleDf.describe().T
```

	count	mean	std	min	25%	50%	75%	max
year	8128.0	2013.804011	4.044249	1983.0	2011.0	2015.000000	2017.0000	2020.0
selling_price	8128.0	638271.807702	806253.403508	29999.0	254999.0	450000.000000	675000.0000	10000000.0
km_driven	8128.0	69819.510827	56550.554958	1.0	35000.0	60000.000000	98000.0000	2360457.0
mileage	8128.0	19.418783	3.981875	0.0	16.8	19.418783	22.2775	42.0
engine	8128.0	1458.625016	497.017504	624.0	1197.0	1248.000000	1582.0000	3604.0
max_power	8128.0	91.517919	35.343246	0.0	68.1	83.100000	101.2500	400.0
seats	8128.0	5.416719	0.946450	2.0	5.0	5.000000	5.0000	14.0

Eksik değerlerin bulunduğu kolonun ortalamasıyla doldurulması

Eksik değerlerle başa çıktıktan sonra, veri setimizin herhangi bir “transformations” gerekip gerekmediğine bakabiliriz.

Sayısal olarak toplanan verilerin ölçekleri çoğunlukla birbirinden farklıdır. Bu farklılığı bazı makine öğrenmesi algoritmaları sevmez ve veri dönüşümü yapılmadan uygulandığında iyi performans gösteremez. Aynı zamanda farklı ölçeklere sahip değişkenleri model sonunda birbiriyle karşılaştırmamız doğru olmaz çünkü iki değişkenin konuşma dili aynı değildir. Benim veri setimde de numerik kolonlarımda hepsi farklı birimlere sahip, bu yüzden standart scaler uygulanabilir. Bir dönüşümü uygulayarak bu değişkenleri aynı metriğe getirmiş oluruz böylelikle artık hem karşılaştırma yapabiliriz hem de algoritmalardan daha iyi performans ede edebiliriz.

Standard Scaler Değişkenleri, ortalaması 0 std sapması 1 olan bir dağılıma çeviriyor . Veri setindeki tüm verilerden ilgili sütun ortalaması çıkartılıp yine sütun std sapmasına bölünerek bulunuyor. Böylelikle veri setindeki tüm gözlem birimleri -1 ile 1 arasında değer almış oluyor. z = (x — u) / s

Starndart Scaler işleminin uygulanması

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
vehicleDf[feature_cols] = scaler.fit_transform(X.values)

vehicleDf
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	0.048464	450000	1.338363	Diesel	Individual	Manual	First Owner	0.999896	-0.423804	-0.495682	190Nm@ 2000rpm	-0.440324
1	Skoda Rapid 1.5 TDI Ambition	0.048464	370000	0.887411	Diesel	Individual	Manual	Second Owner	0.432289	0.079227	0.339607	250Nm@ 1500-2500rpm	-0.440324
2	Honda City 2017-2020 EXi	-1.929775	158000	1.241098	Petrol	Individual	Manual	Third Owner	-0.431678	0.077215	-0.382499	12.7@ 2,700(kgm@ rpm)	-0.440324
3	Hyundai i20 Sportz Diesel	-0.940656	225000	1.011202	Diesel	Individual	Manual	First Owner	0.899435	-0.126009	-0.042951	22.4 kgm at 1750-2750rpm	-0.440324
4	Maruti Swift VXI BSIII	-1.682495	130000	0.887411	Petrol	Individual	Manual	First Owner	-0.833524	-0.323198	-0.093883	11.5@ 4,500(kgm@ rpm)	-0.440324
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8123	Hyundai i20 Magna	-0.196816	320000	0.710567	Petrol	Individual	Manual	First Owner	-0.230756	-0.526422	-0.245265	113.7Nm@ 4000rpm	-0.440324
8124	Hyundai Verna CRDi SX	-1.682495	135000	0.869726	Diesel	Individual	Manual	Fourth & Above Owner	-0.657716	0.069167	0.522963	24@ 1,900-2,750(kgm@ rpm)	-0.440324
8125	Maruti Swift Dzire ZDI	-1.187935	382000	0.887411	Diesel	Individual	Manual	First Owner	-0.029833	-0.423804	-0.498511	190Nm@ 2000rpm	-0.440324
8126	Tata Indigo CR4	-0.196816	290000	-0.792605	Diesel	Individual	Manual	First Owner	1.042592	-0.126009	-0.608864	140Nm@ 1800-3000rpm	-0.440324
8127	Tata Indigo CR4	-0.196816	290000	-0.792605	Diesel	Individual	Manual	First Owner	1.042592	-0.126009	-0.608864	140Nm@ 1800-3000rpm	-0.440324

```
vehicleDf.describe()
```

	year	selling_price	km_driven	mileage	engine	max_power	seats
count	8.128000e+03	8.128000e+03	8.128000e+03	8.128000e+03	8.128000e+03	8.128000e+03	8.128000e+03
mean	-1.378422e-16	6.382718e+05	2.985910e-17	1.733084e-16	1.148059e-17	-6.933430e-17	1.349670e-16
std	1.000062e+00	8.062534e+05	1.000062e+00	1.000062e+00	1.000062e+00	1.000062e+00	1.000062e+00
min	-7.617213e+00	2.999900e+04	-1.234697e+00	-4.877094e+00	-1.679370e+00	-2.589563e+00	-3.610257e+00
25%	-6.933756e-01	2.549990e+05	-6.157615e-01	-6.577164e-01	-5.264223e-01	-6.626262e-01	-4.403242e-01
50%	2.957441e-01	4.500000e+05	-1.736520e-01	-5.898606e-15	-4.238039e-01	-2.381908e-01	-4.403242e-01
75%	7.903040e-01	6.750000e+05	4.983545e-01	7.179764e-01	2.482459e-01	2.753759e-01	-4.403242e-01
max	1.532144e+00	1.000000e+07	4.050850e+01	5.671350e+00	4.316763e+00	8.728713e+00	9.069475e+00

4- Feature Engineering

Korelasyon matrisi ile sütunların birbirleriyle olan ilişkisini gözlemleyebiliriz.

çizdirdiğimiz heatmap'e bakarak "selling_price" ve "max_power" özellikleri arasında yüksek korelasyon olduğu gözüküyor. Fiyat tahmin regresyonum için max_power özelliğinin benim için önemli olduğunu gösterir. Korelasyon sonucunda birbiriyle yakın korelasyona yakın olan özelliklerde bir eleme yapılabilir fakat az sayıda, sadece 6 tane numerik özelliğe sahip olduğum herhangi bir değişiklik yapmayacağım.

Verimi okudum, EDA adımlarını gerçekleştirdim, bilgi sahibi oldum, eksik değerleri doldurup ve transform yaptım, sonrasındaysa özelliklerimin labelım ile olan ilişkisini korelasyon matrisi ile inceledim. Veri setimdeki özellikleri kullanarak bir tahmin işlemi

yapacağım için regresyon gerçekleştireceğim, en iyi sonucu alabilmek için makine öğrenme algoritmalarını karşılaştırıp, parametlerini tune etmem gerekiyor.

5- Machine Learning Problem

Benim seçtiğim machine learning problemi regresyon, regreson ile veri setimdkei araba özelliklerini kulanaraka satış tahmini yapmak.

Regresyon modelinin performansını değerlendirmek için kullanılan metrikler;

[MSE] Ortalama Kare Hata (Mean Squared Error): Basitçe, ortalama kare hata bir regresyon eğrisinin bir dizi noktaya ne kadar yakın olduğunu söyler. MSE, bir makine öğrenmesi modelinin, tahminleyicinin performansını ölçer, her zaman pozitif değerlidir ve MSE değeri sıfıra yakın olan tahminleyicilerin daha iyi bir performans gösterdiği söylenebilir.
Ortalama kare hata ME'de olduğu gibi gerçek değerler ile modelden çıkan değerler arasındaki farkın karesidir. Bu farkın karesinin alınmasının iki sebebi bulunmaktadır. Bunlardan ilki pozitif yönde değer elde etmek ikincisi ise hatanın büyüklüğünü kareli ifadeler ile gösterebilmek. Diğer metriklerde olduğu gibi sıfıra yaklaşması modelin doğruluğunu güçlendirmektedir.

$$MSE\,(E) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

[RMSE] Kök Ortalama Kare Hata (Root Mean Square Error): Bir makine öğrenmesi modelinin, tahminleyicinin tahmin ettiği değerler ile gerçek değerleri arasındaki uzaklığın bulunmasında sıklıkla kullanılan, hatanın büyüklğünü ölçen kuadratik bir metriktir. RMSE tahmin hatalarının (kalıntıların) standart sapmasıdır. Yani, kalıntılar, regresyon hattının veri noktalarından ne kadar uzakta olduğunun bir ölçüsüdür; RMSE ise bu kalıntıların ne kadar yayıldığının bir ölçüsüdür. Başka bir deyişle, verilere en iyi uyan çizgi etrafında o verilerin ne kadar yoğun olduğunu söyler. RMSE değeri 0'dan ∞'a kadar değişebilir. Negatif yönelimli puanlar yani daha düşük değerlere sahip tahminleyiciler daha iyi performans gösterir. RMSE değerinin sıfır olması modelin hiç hata yapmadığı anlamına gelir. RMSE, büyük hataları daha fazla cezalandırmanın avantajına sahiptir, bu yüzden bazı durumlara daha uygun olabilir. RMSE, birçok matematiksel hesaplamada istenmeyen mutlak değerin kullanılmasını engeller.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

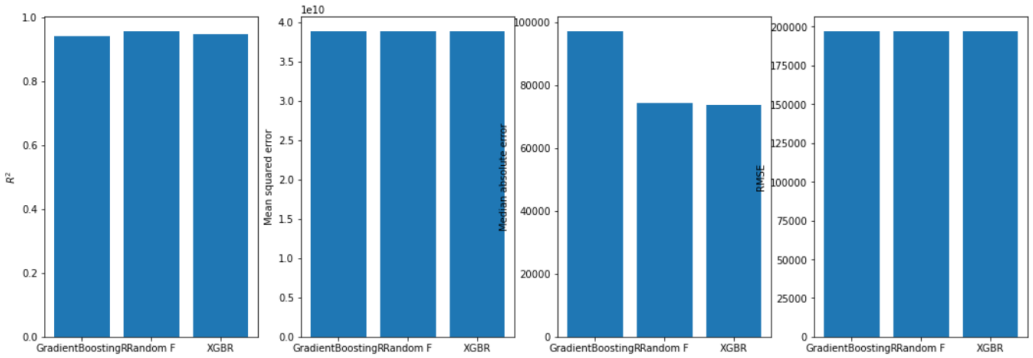
Root Mean Squared Error RMSE (Karekök Ortalama Kare Hata) Yukarıda bahsedilen MSE felsefesinin karekökünün alınmış halidir. Bu sayede kareler ile ortadan kaldırılan matematiksel riskler karekökü alınarak gerçek sonuçların yorumlanmasına katkıda bulunur.

R Square-R^2 (R-Kare) bağımsız bir değişken veya bir regresyon modelindeki değişkenler tarafından açıklanan bağımlı bir değişkenin varyans oranını temsil eden istatistiksel bir ölçüdür. Bu ölçü ile bağımlı değişkenin açıklama oranı verilir. 0 ile 1 arasında olup 1'e yakın oldukça başarı oranı yüksek anlamına gelir.

6- Algorithms, Implementation and Performance Comparison

Regresyon için birçok machine learning algoritması mevcur, bunlardan hangisinin en iyi sonuç vereceğini birbirlerini pipeline ve gridsearch yöntemleri ile aynı ayna karşılaştırarak seçebiliriz.
Ben GradientBoostingRegressor(), RandomForestRegressor(), XGBRegressor() regresyon modellerini seçtim ve MSE, RMSE, ME, R2 değerlerini hesaplayarak karşılaştırdım;

	model	best_score	best_params
0	xgb_regression	0.959392	{'gamma': 0.01, 'learning_rate': 0.1, 'max_dep...
1	gradientboosting_regression	0.956050	{'criterion': 'mse', 'learning_rate': 0.25, 'l...
2	RandomForest_Regression	0.959685	{'criterion': 'mse', 'max_depth': 9, 'n_estima...



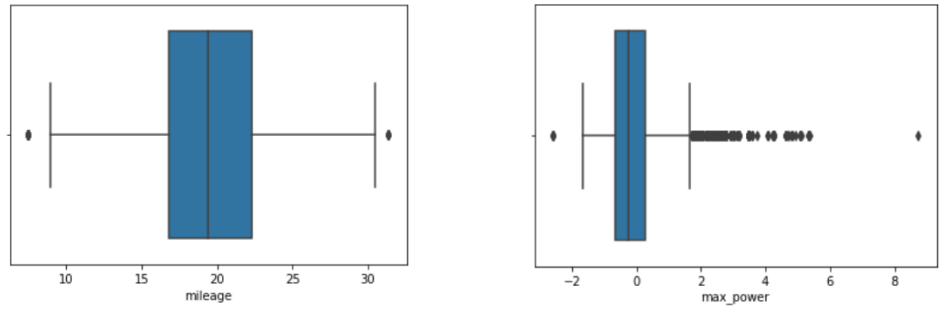
R2 değerimizin en yüksek, hata değerlerimi gösteren mse, rmse, me değerlerinin de en düşük olan model bizim için en iyi modeldir. Tabloya baktığımımz zaman birbirlerine yakın iki modelim olsa da Random Forest modeli daha iyi sonuç verdi.
Bu 3 modelimi daha fazla özelliklerini karşılaştırarak hani parametreleri tune edilmeli bunları inceledim

```
Best parameter (score=0.963):
{'clf__estimator': RandomForestRegressor(n_estimators=200), 'clf__estimator__criterion': 'mse', 'clf__estimator__n_estimators':
200, 'scaler': StandardScaler()}
```

Önceki karşılaştırmamızda da Random Forest modeli bize en iyi sonucu vermişti zaten bu sefer tune edilmesi gereken parametleri arttırarak karşılaştırınca biraz daha iyi bir sonuç alaral 0.963 bir başarı elde ettim.

7- Further Performance Improvement (Your best algorithm)

Veri setimizde aykırı değerler bulunabilir, bunları boxplot çizdirerek görebiliriz, bazı özelliklerimdeki aykırı değerleri görmek için çizdirdim;



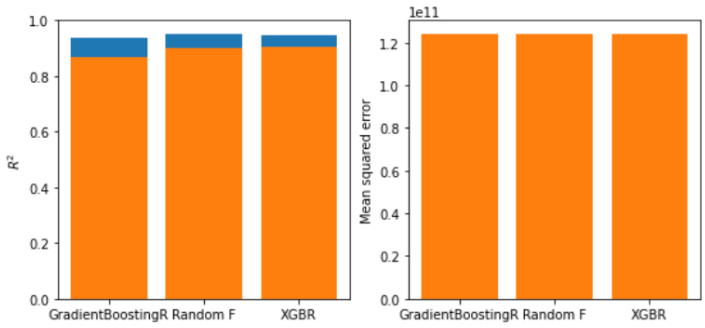
Veri setimdeki aykırı değeleri temizlemek için q1,q3ve iqr değerlerini hesaplayıp daha sonrasında bunları sildim, veri setimde çok fazla aykırı değerler bulunduğu için ciddi bir azalma oldu ve 8128 x 13 iken 5639x13'e kadar düştü.

```
Q1 = copy_df.quantile(q=.25)
Q3 = copy_df.quantile(q=.75)
IQR = Q3 - Q1

data_clean = copy_df[~((copy_df < (Q1-1.5*IQR)) | (copy_df > (Q3+1.5*IQR))).any(axis=1)]

data_clean.shape
```

```
(5639, 13)
```

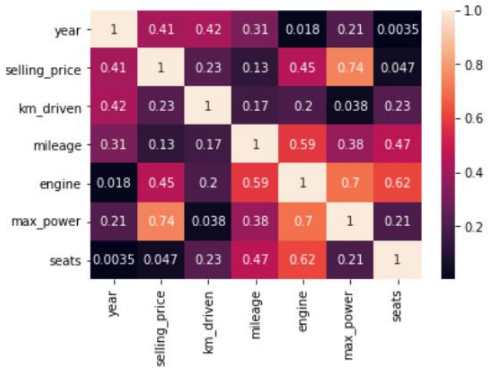
Aykırı değerlerimi sildikten sonra performansına etkisini görebilmek tekrardan 3 model için performanslarını hesapladım

	model	best_score	best_params
0	xgb_regression	0.914561	{'gamma': 0.01, 'learning_rate': 0.1, 'max_dep...
1	gradientboosting_regression	0.904085	{'criterion': 'mse', 'learning_rate': 0.5, 'lo...
2	RandomForest_Regression	0.894860	{'criterion': 'mse', 'max_depth': 9, 'n_estima...



Aykırı değerleri çıkardıktan sonra R2 değerimin düştüğü MSE değerimin azaldığı gözükmekte, buda performansa olan etkisinin olumsuz yönde olduğunu gösterir.
Çok fazla aykırı değerim olduğu için ve hepsini sildiğim için başarı oranım düştü, ayrıca aykırı değerlerimi silmeden önce en iyi sonucu Random Forest'da alırken, aykırı değerlerimi temizledikten sonra en iyi sonucu xgb_regression modeli vermiş oldu.

Başka bir performans değerlendirmesi olarak **“feature selection”** uygulanabilir. Bunun için birkaç yöntem var ilk olarak korelasyon matrisini çizdirerek ve değerlerini hesaplayarak yapılabilir




```
vehicleDf.corr().abs()['selling_price'].nlargest(5)
```

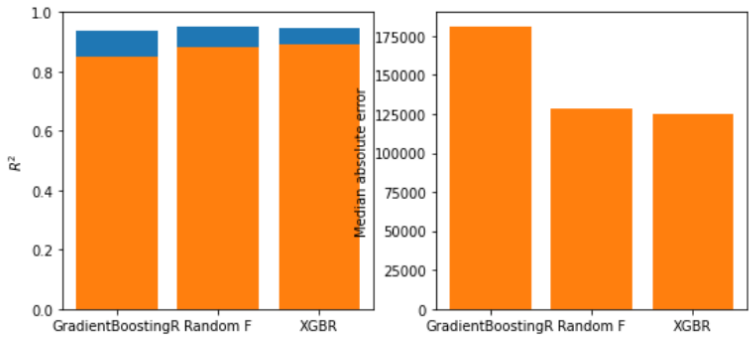
```
selling_price    1.000000
max_power        0.744958
engine           0.453567
year             0.414092
km_driven        0.225534
Name: selling_price, dtype: float64
```

mutlak değerde labelıma (“selling_price”) göre en yüksek korelasyon olan özelliklerim

Birbiriyle yüksek koralasyona sahip veriler zaten aynı şeyi verir bunlardan herhangi birini atabiliriz, sellin_price labelım ile en yüksek korolasyona sahip olan özelliğim max_power, sonrasında engine ve year ikiside çok yakın korelasyona sahip ikisinden birini atabilirim.

Başka bir feature selection yöntemi olarakta SelectFromModel ile kaç özellik seçilmesini gösteren yöntemi kullanarak (8128,2) sonucunu elde ettim bu da 2 özelliğin yeterli olduğunu gösterir bize. Bu sonuca performansı tekrar hesaplayarak karşılaştırdım

	model	best_score	best_params
0	xgb_regression	0.895865	{'gamma': 0.01, 'learning_rate': 0.01, 'max_de...
1	gradientboosting_regression	0.888420	{'criterion': 'mse', 'learning_rate': 0.5, 'lo...
2	RandomForest_Regression	0.888244	{'criterion': 'mse', 'max_depth': 9, 'n_estima...



Özellik seçimini çok fazla azalttığım için, performans olarak daha kötü sonuçlar elde ettim

Aykırı değerleri sildiğimde veya “feature selection” metodu uyguladığım zaman performansımda hiçbir iyileşme olmadı, hatta daha da kötü etkiledi. Performansımı arttırabilmek için şu an yaprığım gibi sadece nümerik değerlerim üzerinden bir regresyon değil de, data setimdeki numerik olmayan değerlerimde de bir transform gerçekleştirerek daha fazla özellik elde edip, daha iyi bir sonuç alabilirdim.

8- Inference Give comments on your findings. Did you achieve your goal in the project?

Projemde verilerimi okudum, analiz ettim, train ve test değerlerimi oluşturudum, ve farklı algoritmalarda deneyerek, hata sonuçlarımı, ve en iyi skorlarımı hesapladım, hesapladığım sonuçlardan ve denediğim algoritmalardan, algoritmalarımın tune etmem gereken parametlererini bulduktan sonra 0.95-0.96 gibi bir sonuç elde ettim, sonrasında aykırı değerlerimi temizleyerek performansıma olan etkisini değerlendirdim fakat daha iyi bir sonuç alamadım, ardından “feature selection” yaparak tekrar performansıma olan etkisine baktım fakat yine daha iyi bir sonuç elde edemedim. En başta hesapladığım sonuçlara göre yorum yapmam gerekirse elde ettiğim 0.95-0.96 değeri bir makine öğrenmesi için iyi bir sonuç, bu yüzden bu projede amacıma ulaştım diyebilirim.

KAYNAKÇA

https://kardelennerdem.com/2022/02/08/veri-on-isleme-adim-3-transformation/

https://ng-dasci.medium.com/feature-scaling-nedir-1fbcd5cd125e

https://veribilimcisi.com/2017/07/14/mse-rmse-mae-mape-metrikleri-nedir/#:~:text=Basit%C3%A7e%2C%20ortalama%20kare%20hata%20bir,iyi%20bir%20performans%20g%C3%B6sterdi%C4%9Fi%20s%C3%B6ylenebilir.