



**KOÇ
ÜNİVERSİTESİ**
GRADUATE SCHOOL OF SOCIAL
SCIENCES AND HUMANITIES

QMBU/INTL 450/550

Advanced Data Analysis in Python

Project Report

Predicting Used

Car Price

Hakan Gündoğdu

049980

1) Introduction

In this data analysis task, I aimed to predict used car price. For this task there were some data sets ready to use, but they are mostly gathered from foreign countries which I do not have enough domain expertise. So that I preferred to scrape data from Arabam.com, which is an online classified publishing platform. Because I know in some countries if the car is painted this increases price of the car, but in Turkey originality of the car is an important aspect; and since I know market dynamics for Turkey better than other countries, I preferred to collect local data.

After data collection, I passed to methodology phase. In this phase I wanted to reach more stable and explainable model. Due to simplicity and being explainable, I started with the linear regression, however standard deviation of the accuracy was high, which means it does not generate stable results. Then to extract meaningful models I used KNN Regression and Gradient Boosting Regression. Since the KNN method works based on distance, to scale the scaler features, I also used normalized and standardized the data to scale up the features. At the end, just see the power of neural network I also run a MLP algorithm, and its accuracy is near to Gradient Boosting and KNN Regression. Since it increases accuracy just a bit, having regards to much more computational effort, I did not prefer to continue with it.

2) Methodology and Road Map









2-1) Data Scraping

I preferred to collect data from an online classifieds and shopping platform, which is Arabam.com. At the beginning I did not want to build a huge dataset, to gain from train and test effort, so that I just collected Volkswagen classifieds given by the dealer.

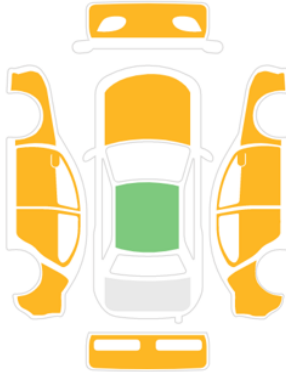
From that site I was able to collect many features, which are mainly, brand name, model name, versions, engine size, originality and more. Features that I collected can be seen below.

	Fiyat	İl	İlçe	İlan No	İlan Tarihi	Marka	Seri	Model	Yıl	Yakıt Tipi	Vites Tipi	Motor Hacmi	Motor Gücü	Kilometre	Boya-değişen	Kimden	Sağ Arka Çamurluk	Arka Kaput	Sol Arka Çamurluk	Sağ Arka Kapsi	Sağ Ön Kapsi	Tavan	Sol
0	91750	ESKİŞEHİR	ODUNPAZARI	17582283	03 Haziran 2021	Fiat	Linea	1.3 Multijet Active Plus	2014	Dizel	Düz	1400	100	164750	Tamamı orijinal	Galeriden	Orijinal	Orijinal	Orijinal	Orijinal	Orijinal	Orijinal	O
1	82750	ESKİŞEHİR	ODUNPAZARI	17545384	03 Haziran 2021	Fiat	Punto	EVO 1.3 Multijet Dynamic	2012	Dizel	Düz	1400	75	198000	Tamamı orijinal	Galeriden	Orijinal	Orijinal	Orijinal	Orijinal	Orijinal	Orijinal	O

Picture 1 Scraped data from the website

Volkswagen Otomobil Galeriden (4.723 Sonuç)								
İlan Sahibi Galeriden x Seçimleri Temizle								
2. El Araçlar	Sahibinden	Galeriden	Yetkili Bayiden	Görünüm:   				
	Model	İlan Başlığı	Yıl	Kilometre	Renk	Fiyat	Tarih	İl / İlçe
	Volkswagen Golf 1.6 Primeline	ERGİN OTOMOTİV'DEN 2005 GOLF	2005	246.000	Lacivert	86.750 TL	30 Mayıs 2021	Kocaeli Körfez
	Volkswagen Passat 1.4 TSI BlueMotion Comfortline	Galeriden Volkswagen Passat 1.4 TSI BlueMotion Comfortline 2014 Model Burdur 	2014	97.000	Gri (Gümüş)	189.000 TL	30 Mayıs 2021	Burdur Bucak
	Volkswagen Jetta 1.6 TDI Comfortline	"Cappa Garage"2014 Volkswagen Jetta 1.6 TDI Comfortline 	2014	195.000	Beyaz	179.500 TL	29 Mayıs 2021	Nevşehir Avanos

Picture 2 List of classifieds

86.750 TL		BOYA - DEĞİŞEN	
KOCAELİ / KÖRFEZ / KUZEY MAHALLESİ			
İlan No:	17598084		
İlan Tarihi:	30 Mayıs 2021		
Marka:	Volkswagen		
Seri:	Golf		
Model:	1.6 Primeline		
Yıl:	2005		
Yakıt Tipi:	LPG & Benzin		
Vites Tipi:	Düz		
Motor Hacmi:	1595 cc		
Motor Gücü:	105 hp		
Kilometre:	246.000 km		
Plaka:	-		
Seri Numarası:	*****		
		<p>Sağ Arka Çamurluk: Boyanmış</p> <p>Sol Arka Çamurluk: Boyanmış</p> <p>Sağ Arka Kapı: Boyanmış</p> <p>Sağ Ön Kapı: Boyanmış</p> <p>Tavan: Orijinal</p> <p>Sol Arka Kapı: Boyanmış</p> <p>Sol Ön Kapı: Boyanmış</p> <p>Sağ Ön Çamurluk: Boyanmış</p> <p>Motor Kaputu: Boyanmış</p> <p>Sol Ön Çamurluk: Boyanmış</p> <p>Ön Tampon: Boyanmış</p> <p>Arka Tampon: Boyanmış</p>	
		<p>● Belirtilmemiş ● Orijinal ● Boyalı</p> <p>■ Lokal Boyalı ■ Değişmiş</p>	

Picture 3 Details of a car

After the first try of scrapping I run a series of feature selection algorithm to see which features make sense. Since the data was just build from one brand name, its affect was zero. Also, engine size, engine power, model name were not making sense, because the versions are specific for that model name and engine specifications. But I wanted to keep those features, because they will help the algorithm if it encounters a newly released version, it will predict that version price by comparing its model name, and engine specifications.

The website was showing maximum 50 classifieds per page and continuing maximum 50 pages, so that to be able collect as much as classifieds, I implemented different links. The link was divided based on, seller information, brand name and price limits and sorted by prices to decrease chance of scrapping any duplicate classifieds. After a run I got lower than I expected

then I suspected from any loss of data and started to investigate and realized that in the source code of the page, table names are differentiating between classifieds released by dealers and owners. Then I implemented one another function to scrape owner releases.

```

get_Price_dealer('https://www.arabam.com/ikinci-el/otomobil/skoda-galeriden?sort=priceTl.desc&take=50&page=') #Skoda_Dealer
get_Price_owner('https://www.arabam.com/ikinci-el/otomobil/skoda-sahibinden?sort=priceTl.desc&take=50&page=') #Skoda_Owner

get_Price_dealer('https://www.arabam.com/ikinci-el/otomobil/hyundai-galeriden?sort=priceTl.asc&take=50&page=') #Hyundai_dealer
get_Price_owner('https://www.arabam.com/ikinci-el/otomobil/hyundai-sahibinden?sort=priceTl.desc&take=50&page=') #Hyundai_desc_Owner
get_Price_owner('https://www.arabam.com/ikinci-el/otomobil/hyundai-sahibinden?sort=priceTl.asc&take=50&page=') #Hyundai_asc_Owner

get_Price_dealer('https://www.arabam.com/ikinci-el/otomobil/fiat-galeriden?take=50&page=') #Fiat_dealer
get_Price_owner('https://www.arabam.com/ikinci-el/otomobil/fiat?&currency=TL&maxPrice=60000&membertype=Sahibinden&sort=priceTl.desc&take=50&page=') #Fiat_max60,000_Owner
get_Price_owner('https://www.arabam.com/ikinci-el/otomobil/fiat?&currency=TL&maxPrice=100000&membertype=Sahibinden&sort=priceTl.desc&take=50&page=') #Fiat_max100,000_Owner
get_Price_owner('https://www.arabam.com/ikinci-el/otomobil/fiat?&membertype=Sahibinden&sort=priceTl.desc&take=50&page=') #Fiat_max150,000_Owner

```

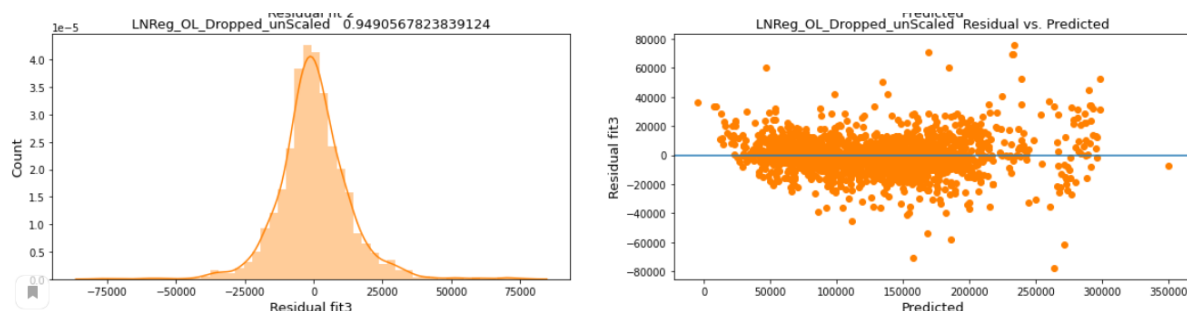
Picture 4 Page links for scraping, divided by price and brand name.

2-2) Methodology

Before I go proceed, I made a literature review, to see how people approach similar problems and found that many people use Multivariate Linear regression for modeling, since it is a simple and an explainable model. It was working good when I just used a one brand name, since the variation was low, but with the new data set, which includes , accuracy is fluctuating between each new train test split. For some splits it worked well and for some splits it failed. For the 8 folded accuracy results of Gradient Boosting regression and Linear Regression results in the table and for a random split, residual distribution is plotted on the graph.

Method	Mean Accuracy	STD Accuracy
GB_Regres_data_unScaled	8.227048e-01	1.655638e-01
LinearRegression_OL_dropped_unScaled	-3.710288e+08	5.570462e+08
LinearRegression_data_unScaled	-5.703643e+08	7.870412e+08
LinearRegression_OL_dropped_standarized	-1.407113e+18	2.913762e+18

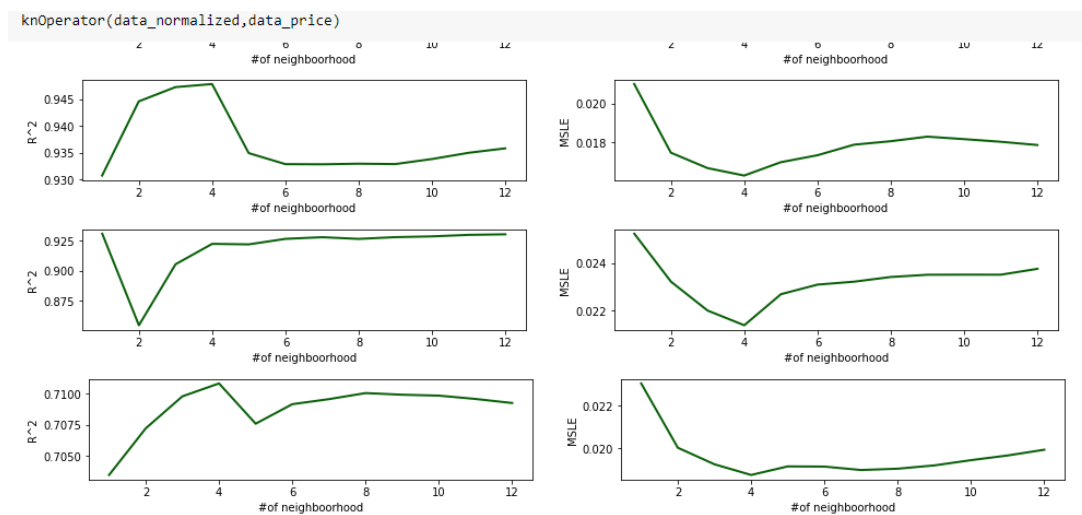
Picture 5 Performance of Linear Regression for both, raw - outliers dropped and scaled - unscaled data sets



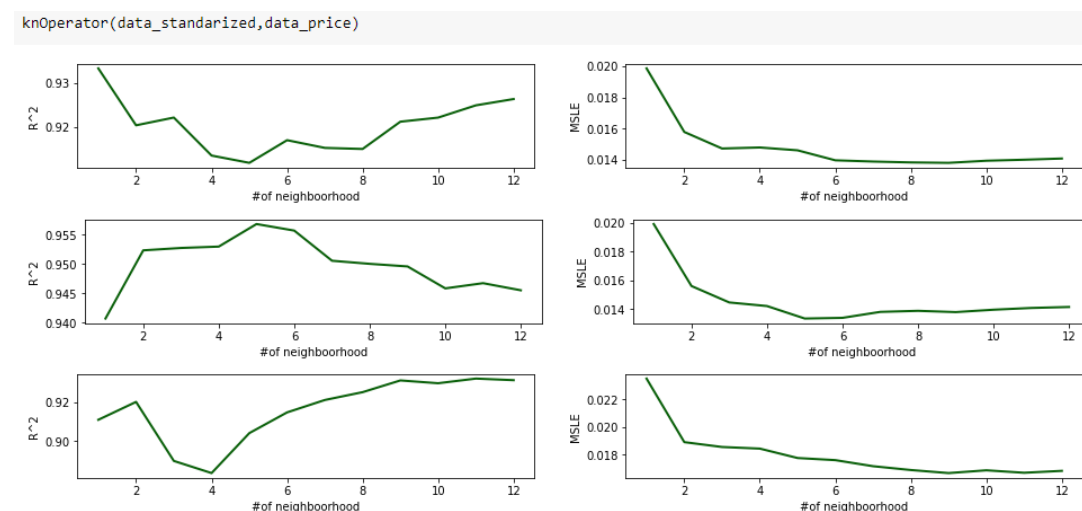
Picture 6 A good Linear Regression result for this problem, but it highly depends on train-test split randomization.

Since linear regression is working unstable for the multiple brand name data set to give more weights to brand, model, and version names, I decided to implement KNN Regression algorithm, which makes calculation considering distances between data and its neighborhoods. So that scaling the numeric features is one of an important aspect for this method. I used normalization and standardization to scale the numeric features and run the algorithm.

Then to decide how many neighborhoods should I check, I implemented a for loop, that creates 10 different test and train splits and runs the algorithm for different number of neighborhoods. Due to working principle of the algorithm, as expected the unscaled data did not worked well, and mostly it converged after the 4th neighborhood with 50% accuracy. However, both normalized and standardized data performed well, and mostly they converged after the 6th neighborhood with more than 90% accuracy.



Picture 7 R^2 and MSLE Plot for different number of neighborhoods for KNN Regression method with the normalized dataset.



Picture 8 R^2 and MSLE Plot for different number of neighborhoods for KNN Regression method with the standardized dataset.

Even if the KNN-Regression method performed better, it is highly possible to fail when we changed the number of neighborhoods, distribution of the accuracy for the different number of neighborhoods can be seen above. Since it is a bit unstable, I kept looking for a new method and decided to use Gradient Boosting Regression.

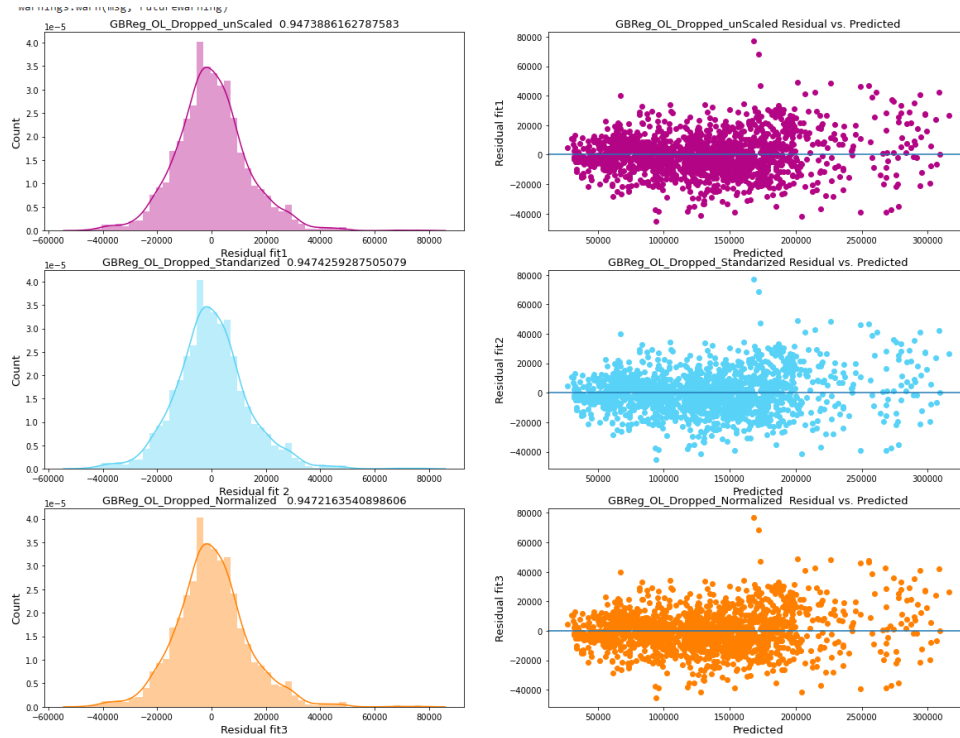
With all experience for this problem, I implemented a series of cross validation test. First, I wanted to see the effect of the outliers, I wanted to use both raw data set, and outliers dropped data set. Then, since accuracy of the methods differentiates between whether they are scaled or unscaled, I created three subsets from those cleared and un-cleared data sets, which means at the end I had six data set to train and test the algorithms.

```
method_scores = {'Method':method, 'Mean Accuracy':method_mean, 'STD Accuracy':method_std
                 }
valuation = pd.DataFrame(method_scores)
valuation = valuation.sort_values(by='STD Accuracy', ascending=True)
valuation
```

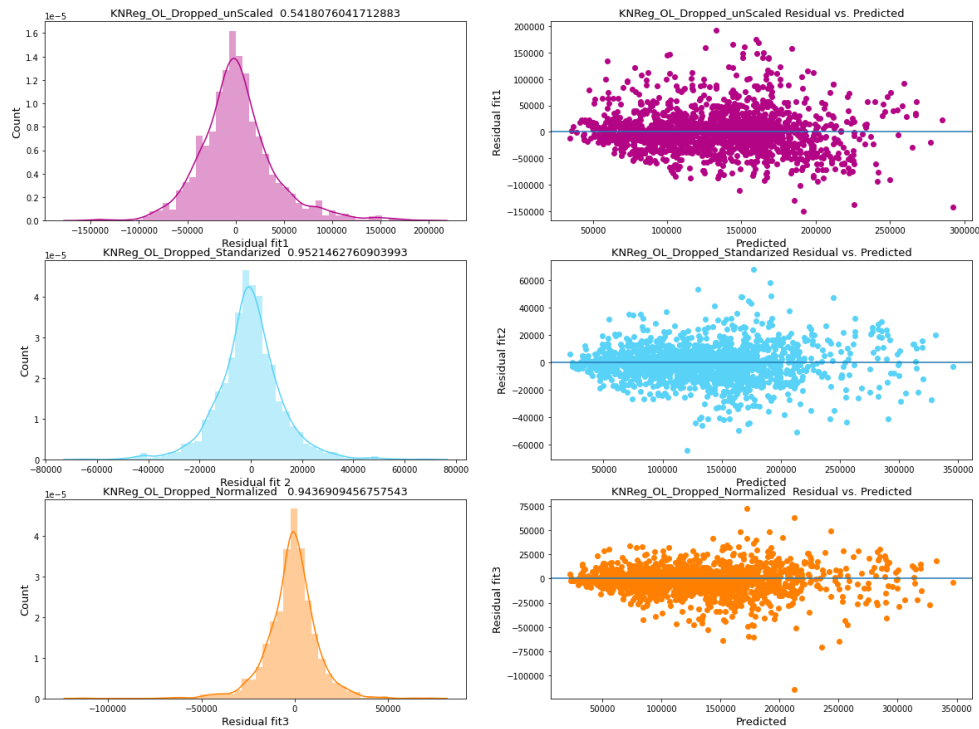
	Method	Mean Accuracy	STD Accuracy
15	GB_Regres_OL_dropped_unScaled	9.498757e-01	5.489028e-03
17	GB_Regres_OL_dropped_standarized	9.496210e-01	5.588333e-03
16	GB_Regres_OL_dropped_normalized	9.497118e-01	5.827837e-03
11	KN_Regres_OL_dropped_standarized	9.421578e-01	6.526606e-03
10	KN_Regres_OL_dropped_normalized	9.353699e-01	8.029111e-03
9	KN_Regres_OL_dropped_unScaled	5.123990e-01	2.789423e-02
0	KN_Regres_data_unScaled	4.831514e-01	7.326247e-02
2	KN_Regres_data_standarized	8.875180e-01	9.140130e-02
1	KN_Regres_data_normalized	8.631720e-01	1.136873e-01
7	GB_Regres_data_normalized	8.318028e-01	1.530920e-01
8	GB_Regres_data_standarized	8.337109e-01	1.606076e-01
6	GB_Regres_data_unScaled	8.227048e-01	1.655638e-01
12	LinearRegression_OL_dropped_unScaled	-3.710288e+08	5.570462e+08
3	LinearRegression_data_unScaled	-5.703643e+08	7.870412e+08
14	LinearRegression_OL_dropped_standarized	-1.407113e+18	2.913762e+18
5	LinearRegression_data_standarized	-4.733853e+18	1.040683e+19
4	LinearRegression_data_normalized	-3.736382e+20	4.145081e+20
13	LinearRegression_OL_dropped_normalized	-1.354517e+22	2.034724e+22

Picture 9 Accuracy results of three methods for both the processed five and the raw dataset

Here when we check the mean and standard deviation of accuracies of eight folded tests, if the outliers are dropped, gradient boosting regression performs well. Then KNN regression comes but, as I discussed before, since its accuracy is sensitive to number of chosen neighborhoods, I prefer to continue with gradient boosting regression.

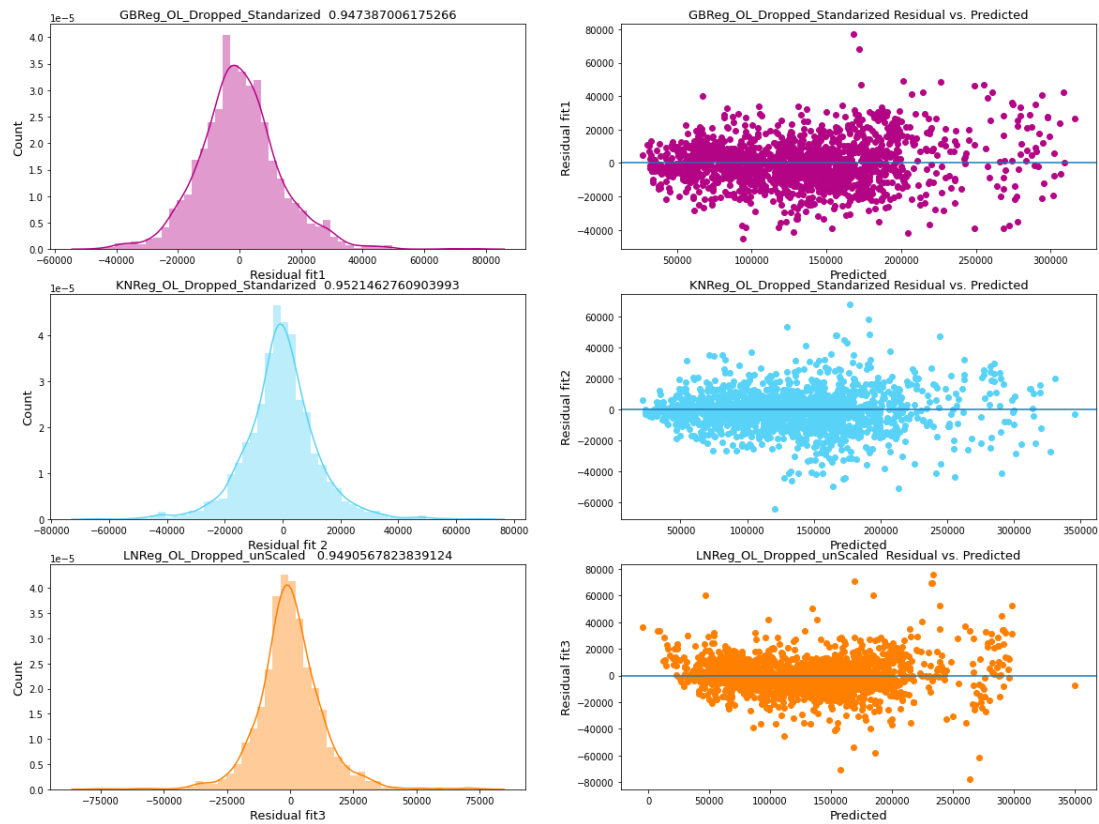


Picture 10 Performance of Gradient Boosting Regression for a chosen train-test split.



Picture 11 Performance of KNN Regression for the same train-test split.

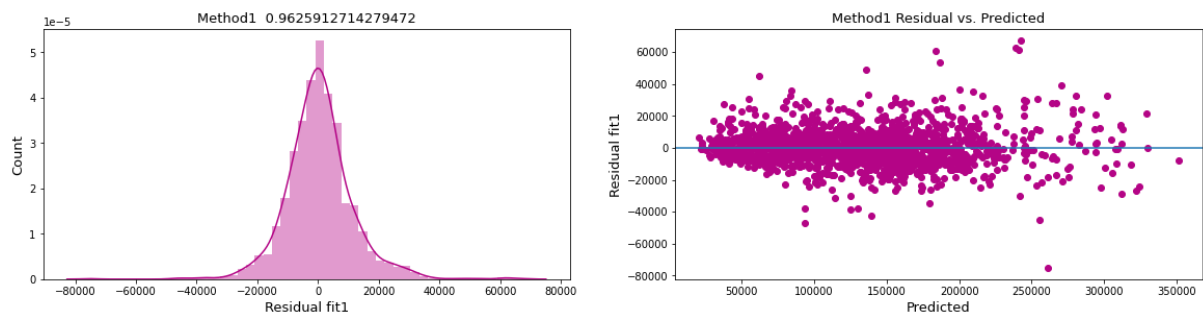
As a chance for this train test split, KNN regression performed better than GB regression, just for fun I also wanted to see performance of the linear regression, the result for both three methods for the same train test split is as follows.



Picture 12 Performance of Linear Regression for the same train-test split.

Linear Regression also performed better than GB Regression, but with the previous observations, considering standard deviations of the methods for this data model, still modelling with GB Regression is preferable.

As a last step I also wanted to see performance of the Multilayer perceptron, for the same train and test split that I used above, MLP returned more than 96% accuracy for the outliers dropped standardized dataset, but its computational effort is a bit high.



Picture 13 Performance of Multilayer Perceptron Regression for the same train-test split.

3) Conclusion

In this study I investigated whether we could predict used car prices or not. To examine this having regard to domain knowledge I preferred to collect car prices from Turkey. After the set is built up, I run four different algorithms to test the hypothesis, whose were Multivariate Linear Regression, KNN-Regression, Gradient Boosting Regression and Multilayer perceptron. Linear regression did not return stable results so that I did not continue to work with it. Then by using KNN-Regression I created a more stable and high accuracy model; however, this model was sensitive to number of neighborhoods that is chosen, which means it stability would change if we changed number of neighborhoods. Then I decided to use Gradient Boosting Regression, it resulted stable and high accuracy models. Mean of R^2 of eight folded test was around 0.95 and standard deviation was not much more than 0.0055. As a last step, to see the power of neural networks, I implemented a Multilayer perceptron and run it. Training time took more than two hours and at the end of training, MLP resulted with 0.96 R^2 value. Considering computational effort to predict used car prices Gradient Boosting Regression would be preferable.