Dataset description:

Data was collected by web crawler form Polish Car trading website - Otomoto.pl
It consists of around 19708 observations (some of them are duplicated) and 18 variables

Objective:

Main objective of this analysis is to build a model which will be able to correctly classify to which Price-group should given car be a part of. Such model can be useful for car buyers or sellers when they are planning to sell or buy a car.

Since variable 'Price' is continuous It was divided into 3 classes 1-highest price 2-medium price and 3-low price car models.

Variables description:

Price-Price of a car
Brand-Brand of a car
Model-Model of a car
Year_produced-year in which car was produced
Mileage-car's mileage
Cylinders_capacity - Car's cylinders capacity
Fuel_type- Diesel/Petrol etc.
HP - Horse Power
transmission - type of transmission
drive_type - type of drive (FWD,AWD,RWD)
Colour-car's colour
Serviced - whether it was serviced in authorized mechanic
New/Used-Describes whether car was used before or is it brand new

```
data_all.columns
[306]  ✓ 0.2s
...  Index(['Price', 'Brand', 'Model', 'Year_produced', 'mileage',
            'Cylinders_capacity', 'Fuel_type', 'HP', 'transmission', 'drive_type',
            'liters_per_km', 'Type', 'CO2 emission', 'No_of_doors', 'No_of_seats',
            'Colour', 'Serviced', 'New/Used'],
           dtype='object')
```

Feature engineering:

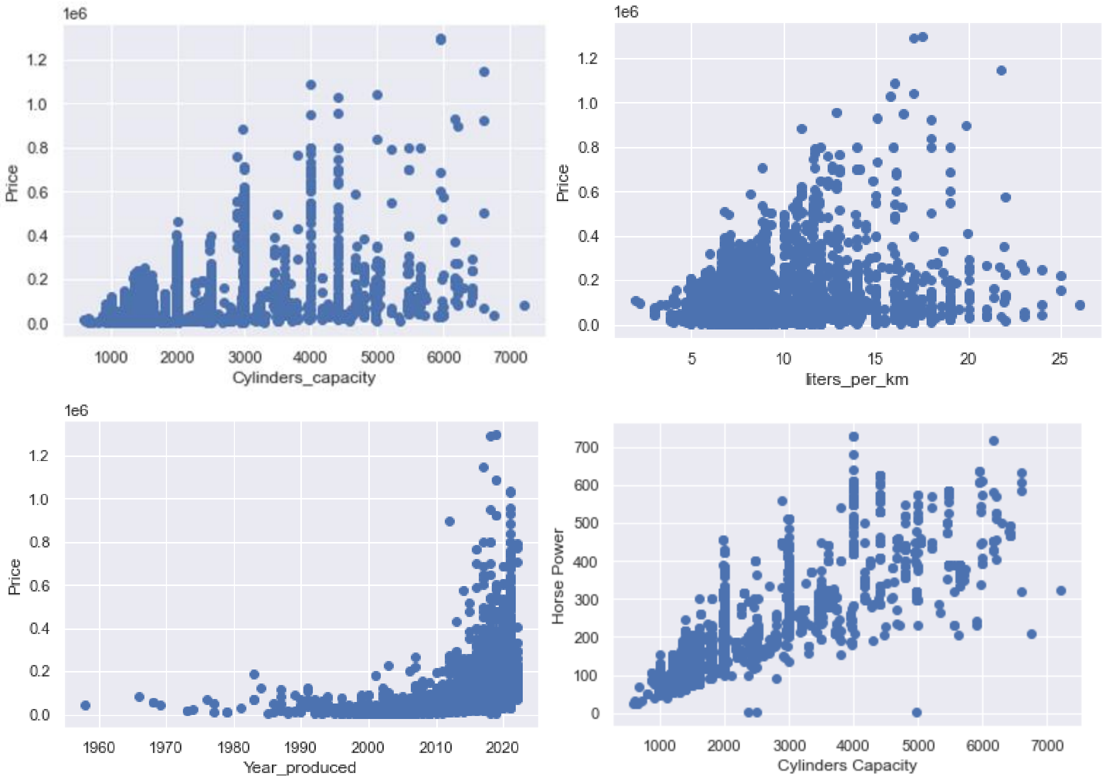
- Removed two outliers (first one because its price was too big and second one because its liters per km ratio was too high)
- Reviewed each variable separately and converted it to an expected type and format
- Filled missing values with mean values of a sub groups
- Removed the rest if there were still missing values
- Grouped car brands and models as ‘Other’ when they appearing too rarely
- All numeric variables were transformed using log1p transformation due to their skewness

Final ‘clean’ dataset consists of 8503 observations

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8503 entries, 1 to 19706
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Price               8503 non-null   float64
 1   Brand               8503 non-null   object
 2   Model               8503 non-null   object
 3   Year_produced       8503 non-null   int32
 4   mileage             8503 non-null   float64
 5   Cylinders_capacity  8503 non-null   float64
 6   Fuel_type           8503 non-null   object
 7   HP                  8503 non-null   float64
 8   transmission        8503 non-null   object
 9   drive_type          8503 non-null   object
 10  liters_per_km       8503 non-null   float64
 11  Type                8503 non-null   object
 12  CO2 emission        8503 non-null   float64
 13  No_of_doors         8503 non-null   float64
 14  No_of_seats         8503 non-null   float64
 15  Colour              8503 non-null   object
 16  Serviced            8503 non-null   object
 17  New/Used            8503 non-null   object
dtypes: float64(8), int32(1), object(9)
memory usage: 1.2+ MB
```

EDA :

Below few of correlation graphs are presented:



All correlations are as we should expect:

-The higher cylinder capacity the higher the price

-The higher the cylinder capacity the more horse power car has

-The higher the fuel burning rate the higher the price

-The newer the car the more expensive it is

Correlation between variables:

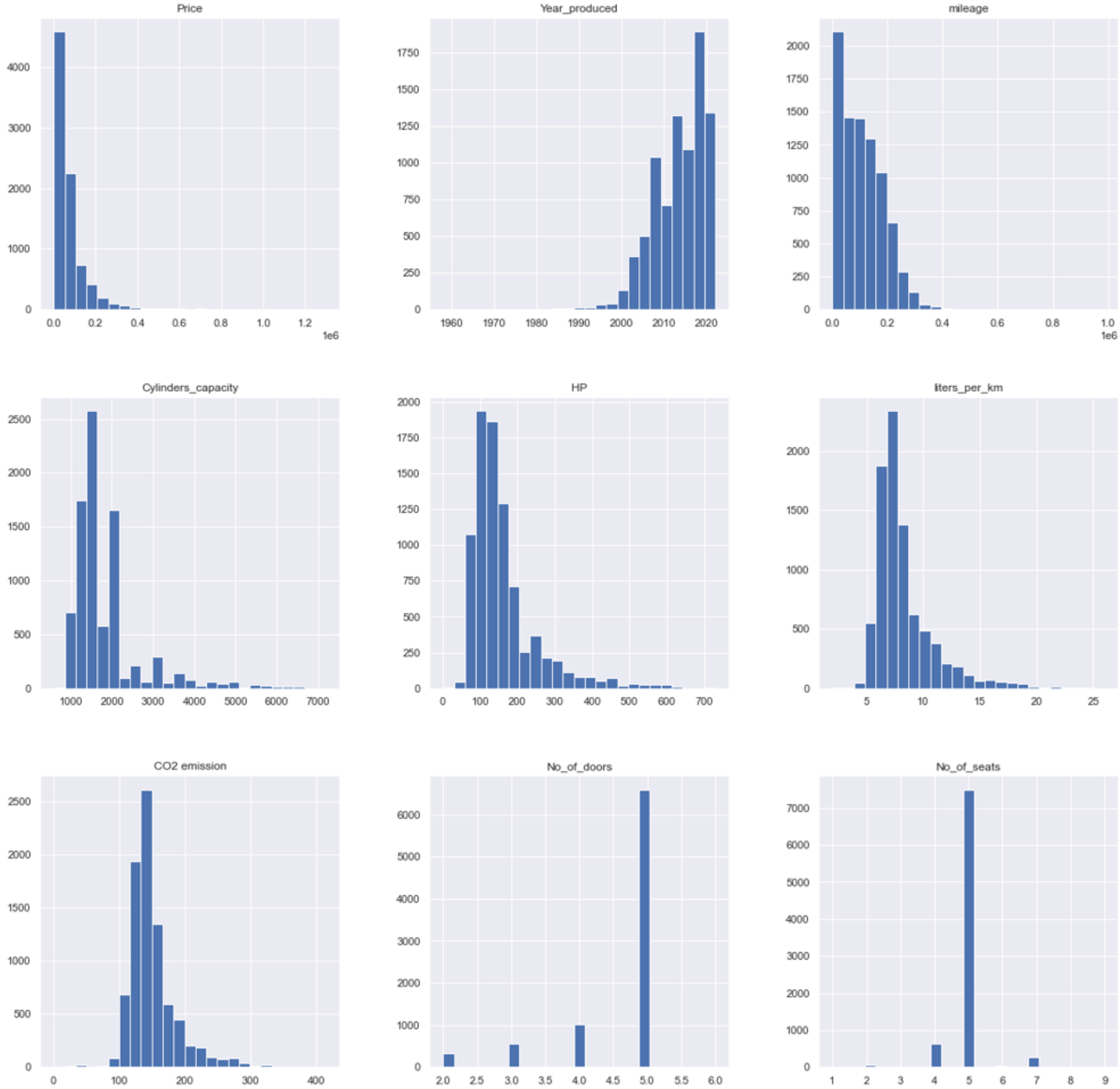
When it comes to direction in which different parameters are correlated with the price:
There are top 30 negatively and positively correlated:

```
mileage                -0.565052
liters_per_km          -0.350799
transmission_Manual    -0.255964
drive_type_RWD         -0.235059
CO2 emission           -0.209621
No_of_doors_3.0        -0.180385
Fuel_type_Petrol       -0.169636
Cylinders_capacity     -0.168949
Colour_Silver          -0.134902
Model_Other            -0.133237
Model_sl               -0.105919
Model_a4               -0.096026
Model_seria3           -0.094904
No_of_seats_4.0        -0.090991
Model_vectra           -0.083228
Colour_Green           -0.073809
Colour_Other           -0.069774
Model_xj               -0.068230
Model_clk              -0.065800
Model_grandvitara      -0.065063
Model_meriva           -0.063083
Colour_Gold            -0.059779
No_of_seats_2.0        -0.059733
Brand_bmw              -0.059520
Model_9-3              -0.059374
Brand_saab             -0.059374
Brand_daihatsu         -0.059018
Brand_honda            -0.058072
Model_klasas           -0.057067
Model_accord           -0.055774
Name: Year_produced, dtype: float64
```

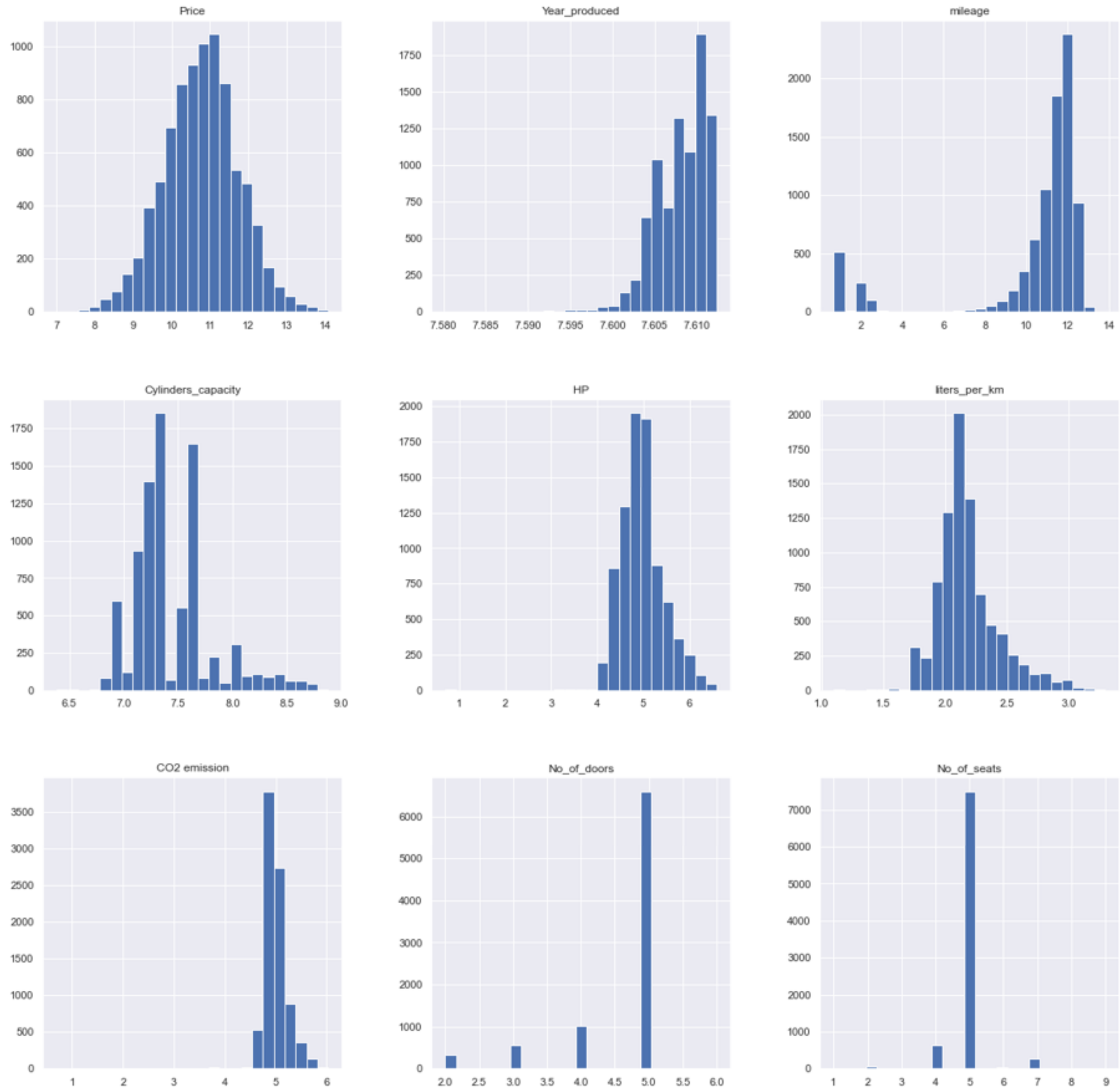
```
Price                0.743590
No_of_doors_5.0      0.220582
Colour_White         0.168758
HP                   0.150132
No_of_seats_5.0      0.118961
Colour_Gray          0.084075
Model_tipo           0.080530
Brand_škoda          0.079569
Model_q3             0.079486
Brand_dacia          0.075577
Model_arteon         0.072460
Model_xc40           0.071079
Brand_hyundai        0.063899
drive_type_FWD       0.063314
Model_kuga           0.059911
Model_stelvio        0.058353
Model_duster         0.056153
Model_tucson         0.056000
Brand_volvo          0.053735
Brand_kia            0.053553
Model_spacestar      0.053518
Model_xc60           0.053138
Serviced_Yes         0.052882
Model_superb         0.052706
Model_compass        0.051220
Model_tiguan         0.049479
Model_giulia         0.049156
Model_ateca          0.048580
Model_captur         0.047847
Name: Year_produced, dtype: float64
```

As we can see all of them are more or less in line with common sense

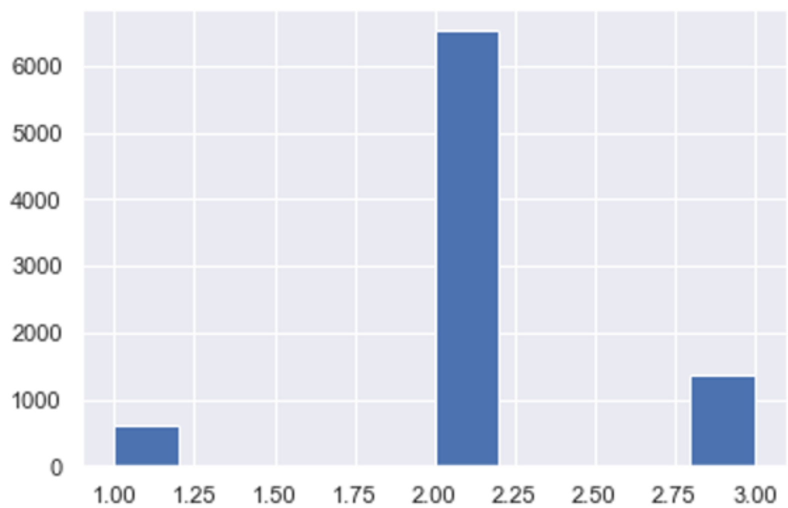
Histograms of continuous variables are presented below, all are indicating high skewness:



Histograms after performing logarithmic transformation look much closer to normal distribution:



Then variable ‘Price’ was transformed into 3 classes <u>1- High price 2-Medium price 3-Low price</u> so that classification models can be used to estimate it:



Classification:

Models used:
-Logistic Regression
-Decision Tree Classifier
-Random Forest Classifier
-Gradient Boosting Classifier

Since classes of Target variable varied greatly in size, my primary metric was ROC AUC

models were fit and train on dataset, training and fitting were done using KFold cross validation, on 5 folds, in order to minimize the overfit some parameters have been introduced :

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,roc_auc_score
from  sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold




model_LR= LogisticRegression(solver='liblinear')
model_tree=DecisionTreeClassifier()
model_forest=RandomForestClassifier(n_estimators=500,random_state=42,warm_start=True)
model_xgb=GradientBoostingClassifier(n_estimators=400, learning_rate=0.009)

param_list={}
✓ 0.5s


kf=KFold(n_splits=5,shuffle=True,random_state=42)
X_train, X_test, Y_train, Y_test = train_test_split(X_class, Y_class, test_size=0.2,random_state=42,shuffle=True)
```

Results:

Logistic Regression

	precision	recall	f1-score	support
1	0.74	0.21	0.33	118
2	0.89	0.97	0.93	1291
3	0.89	0.80	0.85	292
accuracy			0.89	1701
macro avg	0.84	0.66	0.70	1701
weighted avg	0.88	0.89	0.87	1701

Decision Tree Classifier

	precision	recall	f1-score	support
1	0.70	0.62	0.66	118
2	0.94	0.95	0.94	1291
3	0.87	0.88	0.88	292
accuracy			0.91	1701
macro avg	0.84	0.81	0.83	1701
weighted avg	0.91	0.91	0.91	1701

Random Forest Classifier

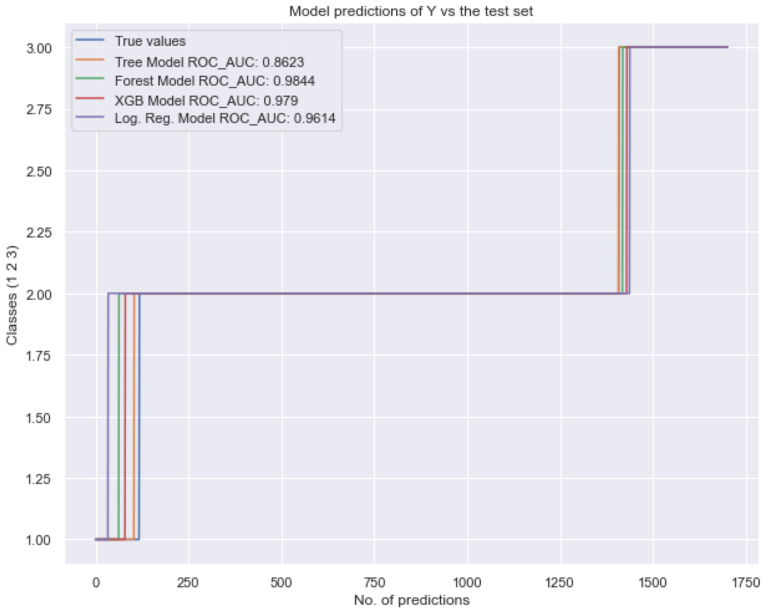
	precision	recall	f1-score	support
1	0.92	0.49	0.64	118
2	0.93	0.98	0.95	1291
3	0.92	0.89	0.90	292
accuracy			0.93	1701
macro avg	0.92	0.79	0.83	1701
weighted avg	0.93	0.93	0.92	1701

Gradient Boosting Classifier

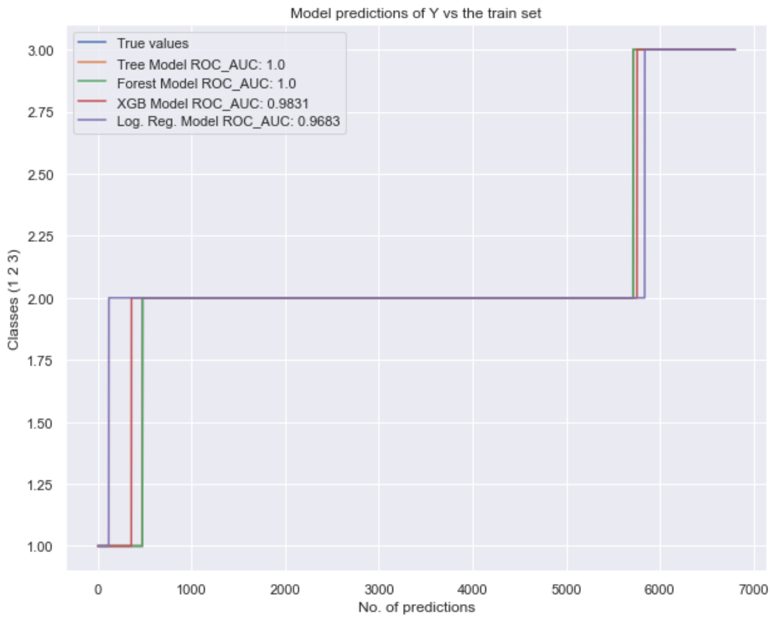
	precision	recall	f1-score	support
1	0.86	0.58	0.70	118
2	0.93	0.98	0.95	1291
3	0.92	0.86	0.89	292
accuracy			0.93	1701
macro avg	0.91	0.81	0.85	1701
weighted avg	0.93	0.93	0.92	1701

When it comes to ability of identifying particular classes Random Forest generally yield better results, however f1 score for class 1 was higher in case of XGB model.

Below is the summary of how those models performed on test set (around 1700 values were used as test set):



And test set (around 700 values were used as a train set):



Best model overall in this case is **Random Forest**. However due to its high overfitting on training set (even after initial tries to reduce it) **XGBoost** may present more stable results. However it can be done as a next step on improving the model. To focus on just those two classifying methods and modify the hyperparameters.