Capstone Project

Machine Learning Engineer Nanodegree

Abdullah Hany

January 31th, 2019

Definition

Project overview

Supervised learning is one of the most promising field in Machine-learning where already much development has taken place and is currently used in real world application.
Supervised learning is a Machine-learning task of teaching a function to map an input to an output based on previous input-output pairs.

Nowadays almost everyone has a car, people tend to change their cars frequently in order to enjoy up to date technology, since cars are not cheap, they tend to sell their old car to gain some money and use it to buy a new car.
The project's goal is to create a regression model that estimates the price of people's car for sale based on the market.

An example of an academic paper where machine learning is applied to this type of problem is, https://www.researchgate.net/publication/319306871_Predicting_the_Price_of_Used_Cars_using_Machine_Learning_Techniques
The source of data that I will be using is https://www.kaggle.com/orgesleka/used-cars-database#autos.csv

Problem statement

The goal is to create a trained model that predicts the price of the used cars based on number of kilometers, year of registration, month of registration, vehicle type, gearbox, model of the car, fuel type, brand and if the car has a damage, which is not repaired yet

The tasks involved are,

- Download the dataset
- Clean the dataset by removing columns that will not affect the price prediction
- Remove the NULL values and duplicates
- Focusing on columns that have strong correlation with the price column
- Using natural log to normalize the data
- Using one-hot encoder to encode categorical features
- Use some visualization to have a better understanding on the data
- Split the data into training and testing sets with ratios 80 to 20 respectively
- Train a regression model to predict the price of the car

The final model is expected to accurately estimate the price of a used car based on the market value

Metrics

The metrics that will be used to for this regression model is R-squared scoring function as it is a statistical measure of how close the data are to the fitted regression line

R-squared = Explained variation / Total variation

R-squared is always between 0 and 100%:

- 0% indicates that the model explains none of the variability of the response data around its mean.
- 100% indicates that the model explains all the variability of the response data around its mean.

Analysis

Data exploration

The data I will be using is consisted of 370,000 used cars information from Ebay-Kleinanzeigen

Which represents the used car market in Germany.

The fields of the dataset that will be used in the problem

- price: the price on the ad to sell the car
- vehicleType: the type of the vehicle
- yearOfRegistration: at which year the car was first registered
- gearbox: the type of the car's transmission
- powerPS : power of the car in PS
- model
- kilometer : how many kilometers the car has driven
- monthOfRegistration : at which month the car was first registered
- fuelType
- brand
- notRepairedDamage : if the car has a damage which is not repaired yet

Fields that will not be used in the problem, as they are not relevant to the price prediction

- dateCrawled : when this ad was first crawled
- seller : private or dealer
- offerType:
- abtest
- dateCreated : the date for which the ad at Ebay was created
- name: the advertising of the car
- nrOfPictures: number of pictures in the ad
- postalCode
- lastSeenOnline : when the crawler saw this ad last online

Preview of the first three data entries in the dataset

dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model
3/24/2016 11:52	Golf_3_1.6	privat	Angebot	480.0	test	NaN	1993.0	manuell	0.0	golf
3/24/2016 10:58	A5_Sportback_2.7_Tdi	privat	Angebot	18300.0	test	coupe	2011.0	manuell	190.0	NaN
3/14/2016 12:52	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800.0	test	suv	2004.0	automatik	163.0	grand

gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen
manuell	0.0	golf	150000	0.0	benzin	volkswagen	NaN	3/24/2016 0:00	0.0	70435.0	4/7/2016 3:16
manuell	190.0	NaN	125000	5.0	diesel	audi	ja	3/24/2016 0:00	0.0	66954.0	4/7/2016 1:46
automatik	163.0	grand	125000	8.0	diesel	jeep	NaN	3/14/2016 0:00	0.0	90480.0	4/5/2016 12:47

Some statistics about the data before cleaning

	price	yearOfRegistration	powerPS	monthOfRegistration
count	3.715380e+05	371537.000000	371538.000000	371537.000000
mean	1.729544e+04	2004.577883	115.548840	5.734473
std	3.587905e+06	92.865496	192.137238	3.712383
min	0.000000e+00	1000.000000	0.000000	0.000000
25%	1.150000e+03	1999.000000	70.000000	3.000000
50%	2.950000e+03	2003.000000	105.000000	6.000000
75%	7.200000e+03	2008.000000	150.000000	9.000000
max	2.147484e+09	9999.000000	20000.000000	12.000000

- Some cars are sold with price zero
- Some cars are overestimated with price 2.147e+9
- 50% of the cars were registered before year 2004
- Some cars have year of registration 9999 which is infeasible
- 50% of the cars have power of 105 or less
- Some cars are exaggerating their power

Some characteristics about the data

- The data was not equally distributed as well as there was a lot of NULL values, duplicates and unreasonable entries.
- The price, year of registration and powerPS had skewed values and unreasonable entries
- In the year of registration column, every car that was registered before 1960 or after 2016 is considered an outlier and was removed
- In the powerPS every car that had horsepower less than 5 or more than 1000 is considered an outlier and was removed

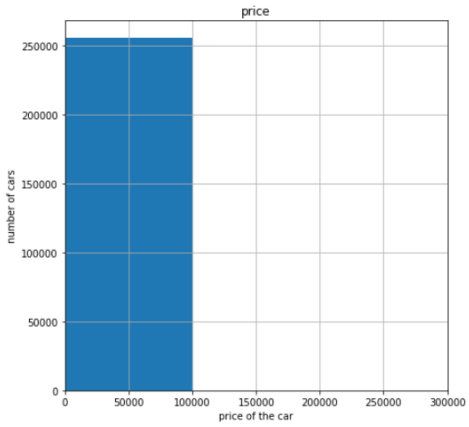
- In the price column all cars that were sold with price equal zero considered outlier and was removed
- In the month of registration column there was a lot of entries with month of registration equal 0 which was considered invalid and removed
- As for the name column, there is no standard convention for entering the name i.e. each person enters the name the way he like, thus making it almost impossible to hot-encode the name column as it will consume a lot of time and space and will not be worth it

Exploratory visualization

The most correlated columns with the price column were year of registration, month of registration and powerPS columns

Histogram graph is used as it helps in visualization of data distribution

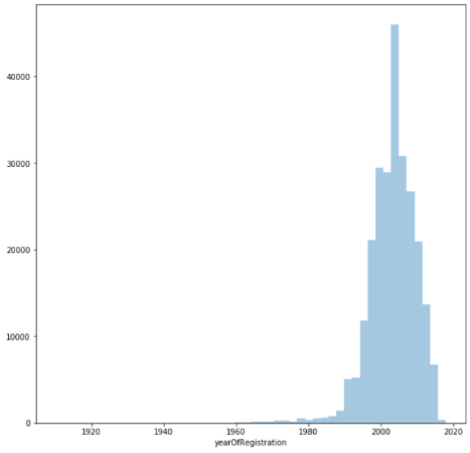
Observing the price column



This histogram graph represents the price column and each car is valued, the histogram shows that most of the cars prices are between 0 and 100000 however, there were 112 cars above the price of 150000 and 3194 cars with the price of 0.

The data needs normalization as the data is skewed

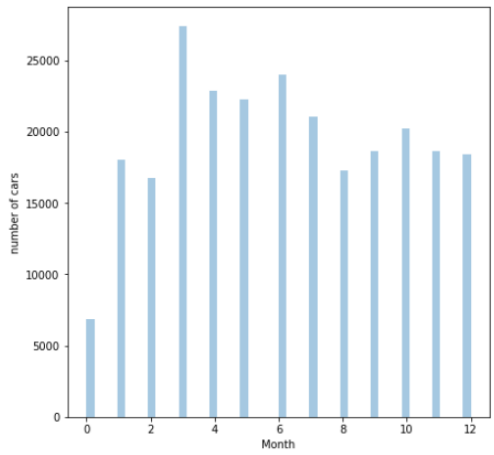
Observing the year of registration column



There was exactly 127 cars registered before 1960 which is considered very old and 10 cars registered after 2016 which is considered very new

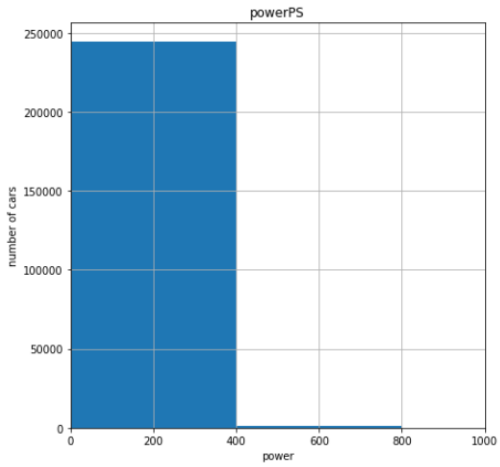
The data needs normalization as the data is skewed

Observing the month of registration column



The cars were equally distributed, however there are 6847 with month of registration 0

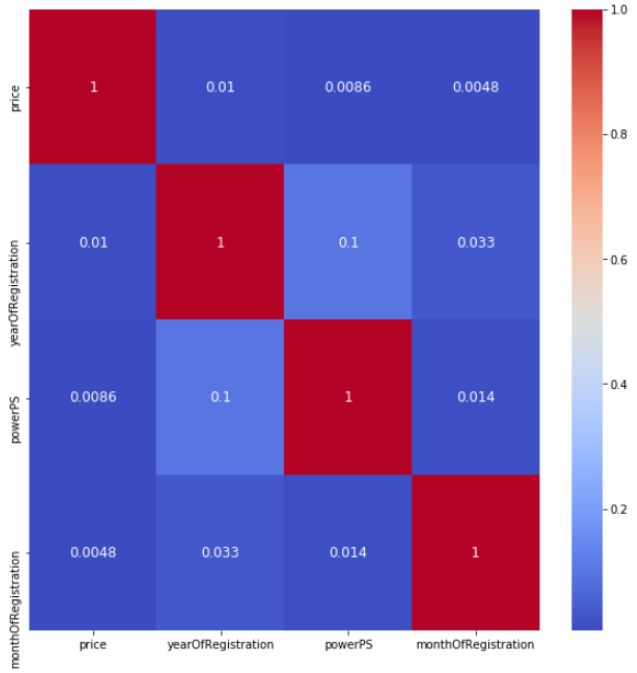
Observing the powerPS column



Some cars has very low power and others have very high power which in both cases is infeasible

Exploring the correlation between the year of registration , powerPS , month of registration and the price column before cleaning the data

	price	yearOfRegistration	powerPS	monthOfRegistration
price	1.000000	0.010340	0.008564	0.004770
yearOfRegistration	0.010340	1.000000	0.100522	0.032599
powerPS	0.008564	0.100522	1.000000	0.014107
monthOfRegistration	0.004770	0.032599	0.014107	1.000000



The heat map graph is used as it helps in visualizing the correlation between the features

Algorithms and Techniques

The main algorithm used to solve this problem was random forest regressor algorithm.

The main idea of random forest is that it splits the data into subsets, train them with decision tree algorithm then combines multiple decision trees to compute the output rather than relying on individual decision trees

The characteristics of the random forest regressor are

- It is powerful and accurate
- Relatively fast compared to other algorithms
- Good performance on many problems
- Does not require all data features to be on the same scale
- handles continuous data

Grid search technique was used to help in parameter tuning,

The idea behind grid search that it tries different parameter combination until it finds the best combination that provides the highest accuracy

Parameters tuned

- n_estimators (the number of trees in the forest)
- criterion (the function to measure the quality of the split)
- min_samples_split(the minimum number of samples required to split an internal node)
- min_samples_leaf(the minimum number of samples required to be at a leaf node)

The secondary algorithm used was a simple linear regression algorithm that acts as a benchmark for the random forest regressor algorithm.

The main idea of linear regression that It tries to accurately fit a line through the data, this fitted line is used to predict values and calculate error.

The characteristics of linear regression are

- very fast
- the data does not need to be normally distributed
- handles continuous data

Benchmark

In this project, I used two benchmarks

The first was trying to beat Kaggle's R-squared score, which was 82.7% using random forest regressor

Link of the kaggle's kernel https://www.kaggle.com/ddmngml/trying-to-predict-used-car-value/notebook

I chose this benchmark because the kernel was using the same dataset and same algorithm that I use.

The second was trying to beat the R-squared score of the linear regression algorithm, which was 74.4%

I chose this benchmark because linear regression is considered a standard approach for regression problems and both random forest and linear regression algorithms handles continuous data and non-linear relationships in data

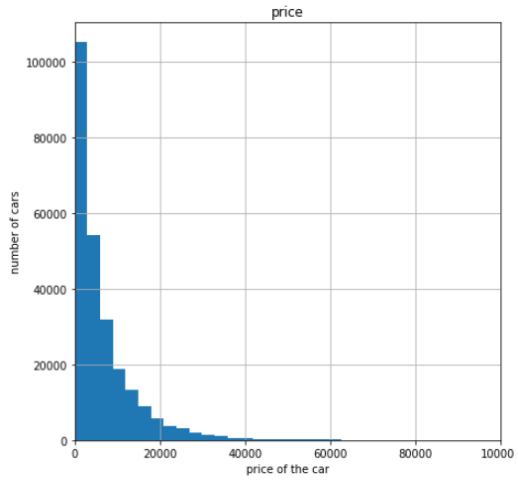
Methodology

Data Preprocessing

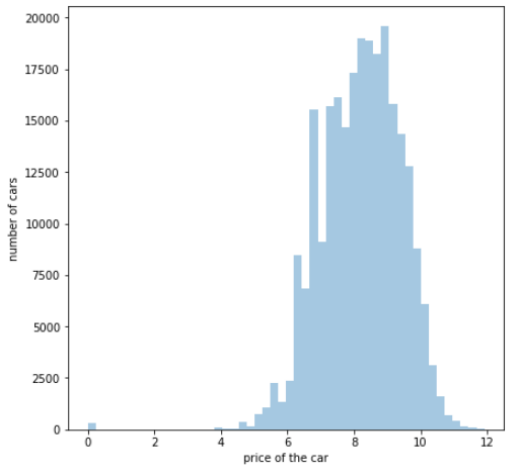
- Some columns were removed, as they have no relevance for the price prediction
    - offerType
    - abtest
    - dateCrawled
    - dateCreated
    - postalCode
    - lastSeen
    - name(advertising name)

- Some columns were removed as they were had only one value
    - seller
    - nrOfPictures

- There was some duplicate values and NaNs which had to be removed
- In the price column there was some overestimation and underestimation in car values so the cars with price more than 150000 were removed , and cars with price equal to 0 were also removed

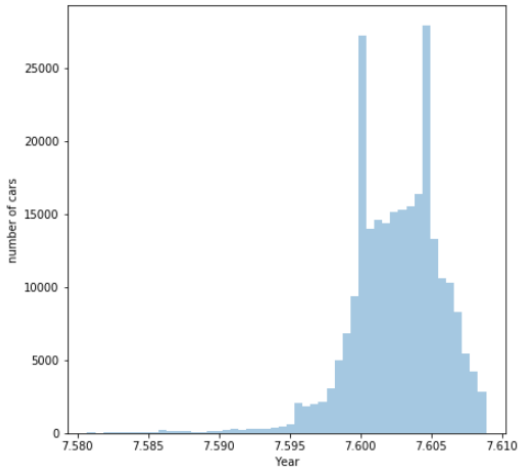


- Normalization on the price column is done using natural log

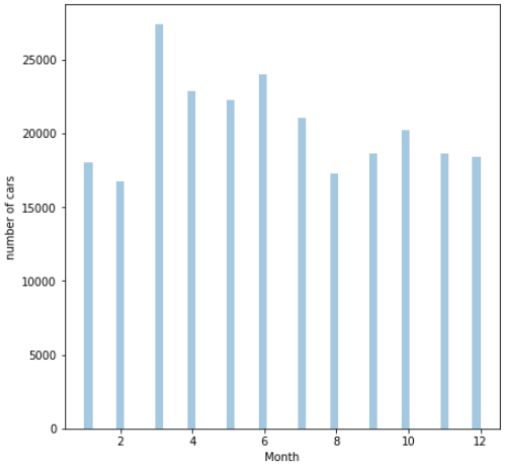


- In the year of registration column every car that was registered before 1960 or after 2016 was considered an outlier and removed

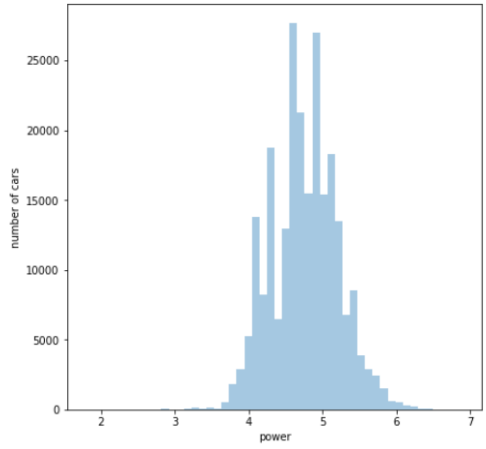
- Normalization on the year of registration column is done using natural log



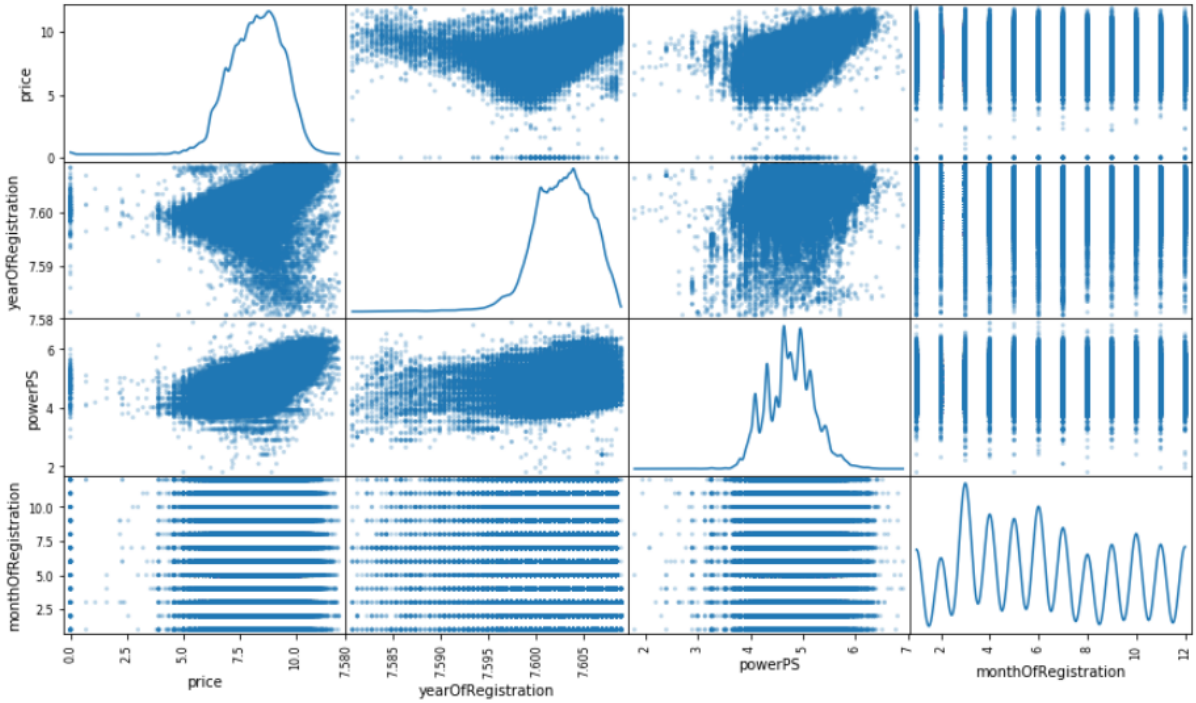
- In the month of registration column some cars were registered at month zero which is infeasible so they were removed



- In the powerPS column some cars had a lot of power and others had very few power, so all cars with power more than 1000 or less than 5 were removed
- Normalization on the powerPS column is done using natural log



- Observing the correlation between the year of registration , powerPS , month of registration and the price column after cleaning the data



Implementation

I implemented three functions to help me in plotting

```
|: # plots seaborn distribution plot
   def distplot(feature,xlabel,ylabel):
       fig, ax = plt.subplots(figsize=(7, 7)) #controls the size of the plot
       ax= sns.distplot(data[feature],kde=False,bins=50) #plot the figure
       ax.set(xlabel=xlabel, ylabel=ylabel)  #controls the label on x and y axis
```

The distplot function helps in plotting the Seaborn distribution plot, the reason I put the distribution plot in a function is to avoid duplicate code

```
def histgram(feature , xlabel,ylabel,lim,bins):
    plt.figure(figsize=(7,7))
    data[feature].hist(bins=bins)
    plt.xlim(0,lim)     #controls the scale on x axis
    plt.title(feature)  #controls the title of the plot
    plt.xlabel(xlabel)  #controls the label on x axis
    plt.ylabel(ylabel)  #controls the label on y axis
    plt.show()
```

The histogram function helps in plotting the histogram plot using matplotlib, the reason I put the histogram plot in a function is to avoid duplicate code

```
# plots all columns except price, yearOfRegistration,powerPS and monthOfRegistration using countplot
def plotting_columns(data,columns):
    for column in columns:
        graph_columns = np.minimum(len(data[column].unique()),5)  #controls the number of bars plotted
        order = data[column].value_counts().iloc[:graph_columns].index #controls the order of attributes plotted
        fig, ax = plt.subplots(figsize=(10, 10))
        sns.countplot(x=column,data=data,order=order,ax=ax).set_title(column) #plots the figure
```

The plotting_columns function helps in plotting all columns except the price, year of registration, powerPS and month of registration

Implementing the linear regression algorithm with the help of sklearn made it very simple,using r2_score to calculate the error between predicted and actual values

linear regression

```
linear_regression =LinearRegression() #intiallizing the model
linear_regression.fit(X_train,y_train) #fitting the model to the training sets
linear_regression_predicted = linear_regression.predict(X_test) #making the model predict the values of the testing set X
```

using r2_score to measure the accuracy of the linear regression model

```
#calculating the error between the values predicted by the model and the actual set
print( r2_score(y_test,linear_regression_predicted))
```

Implementing unoptimized random forest with the help of sklearn

```
random_forest_model = RandomForestRegressor(random_state=42)  #intiallizing the model
random_forest_model.fit(X_train,y_train)#fitting the model to the training sets
random_forest_predicted =random_forest_model.predict(X_test) #making the model predict the values of the testing set X
```

using r2_score to measure the accuracy of the unoptimized model

```
#calculating the error between the values predicted by the model and the actual set
print( r2_score(y_test,random_forest_predicted))
```

Implementing random forest with the help of gridsearch technique in parameter tuning

```
random_forest_model = RandomForestRegressor(random_state=42)
#set of the parameters that the grid search technique will create all possible combinations from
parameters = {
     "n_estimators": [10,100],
    "criterion" : ["mse"],
    'min_samples_leaf':[1,2],
    'min_samples_split':[2,4]

}
#intiallizing the gridsearch object with the random forest model and setting verbose to print the process steps
grid_obj = GridSearchCV(random_forest_model, parameters,verbose=2)
#fitting the gridsearch object with the training sets
grid_fit = grid_obj.fit(X_train, y_train)
#getting the best model
best_model = grid_fit.best_estimator_
#letting the best predict the values of X_test
best_predictions = best_model.predict(X_test)
```

using r2_score to measure the accuracy of the random forest

```
#calculating the error between the values predicted by the model and the actual set
print("Final r2_score on the testing data: ",r2_score(y_test, best_predictions))
```

optimized parameters

```
#getting the best parameters that have the highest accuracy
best_params = grid_obj.best_params_
print(best_params)
```

Refinement

I was able to improve the random forest regression algorithm with the help of grid search technique in parameter tuning.

The list of parameters that grid search helped in tuning are

- n_estimators with values either 10 or 100
- criterion with value of ‘mse’
- min_samples_leaf with values of 1 or 2
- min_samples_split with values of 2 or 4

The grid search technique was able to improve the R-squared score of the random forest regression algorithm from 82.2 to 84.4 with parameters that produced highest accuracy

{'criterion': 'mse', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}

Results

Model evaluation and validation

The final model to solve this problem is the random forest regression model with parameters

{'criterion': 'mse', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}

That were tuned using grid search technique and was able to provide a R-squared accuracy of 84.4%

The method used for model evaluation and validation is R-squared so that I can compare my results with kaggle's benchmark

The random forest regression model was chosen because of

- It is powerful and accurate
- Relatively fast compared to other algorithms
- Does not require all data features to be on the same scale
- handles continuous data

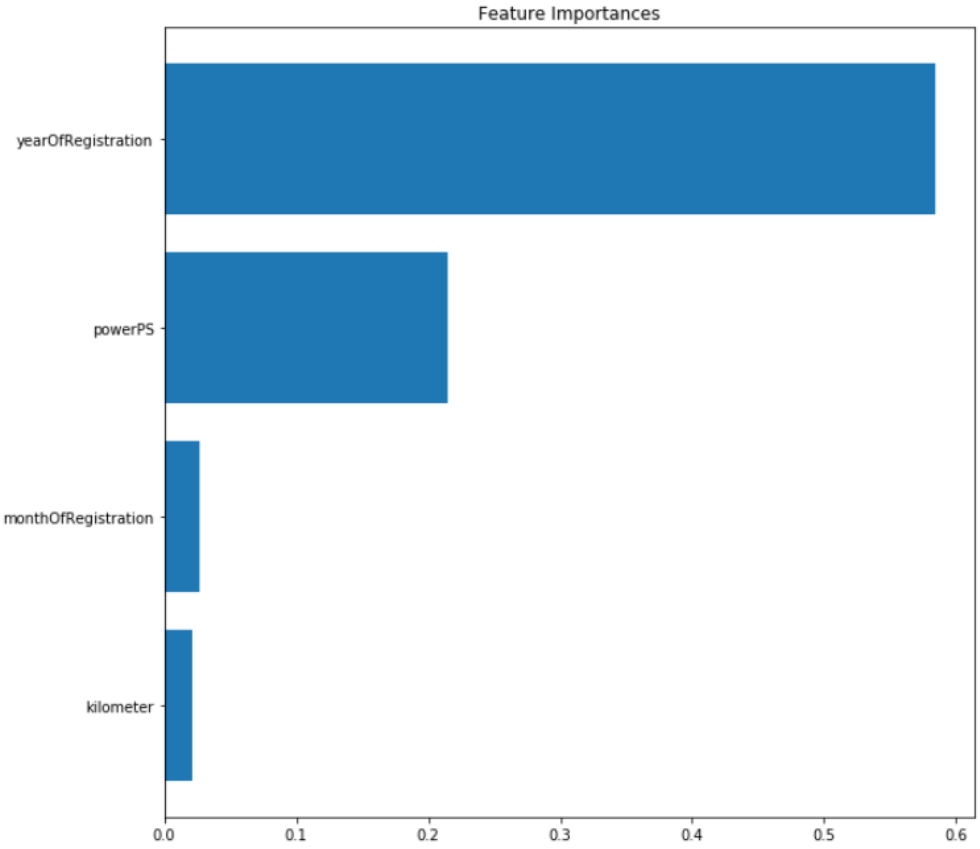
Justification

I was able to beat the linear regression benchmark that had R-squared score of 74.4%

And Kaggle's R-squared accuracy which was 83%, with my random forest regression R-squared score of 84.4%

In my opinion, the R-squared score is significant enough to solve the problem as the R-squared score represents the ability of the model to correctly predict the price of used cars with an accuracy of 84.4%

Free-Form Visualization



For the random forest model, the most important features are year of registration, powerPS, month of registration and kilometer

Reflection

The process of solving this problem can be summarized in a few steps

- Getting the dataset
    - the dataset for this project was retrieved from Kaggle
- Data exploration
- The use of visualization to better understand the data
- Data cleaning and preprocessing
- Splitting the data into testing and training sets
- Building the benchmark model and evaluate it
- Building the proposed model
- Optimizing and tuning the parameters of the proposed model
- Model validation
- Viewing important features

The interesting thing I found that the year of registration has such an influence on the price of the car even more than the power car or the number of kilometers driven.

We can say that the most difficult part was the data cleaning; people tend to submit unreasonable values, which needed to be thoroughly checked and dealt with

Improvement

Improvements that can be done in the future

- Trying different algorithms but with the help of an accelerated GPU as this data is large and consumes time
- Grid search with more values on an accelerated GPU
- Combining more than one model into a custom ensemble model
- Using deep learning