MACHINE LEARNING

UE17EC337

PROJECT REPORT

on

Car Resale Value Prediction

Under the guidance of

Prof. Vanamala H R

Submitted by:

ASHWIN UMADI (PES1201701517)
RAHUL R K (PES1201701552)
SHREYAS KULKARNI (PES1201700450)

Index

Introduction

With the increasing number of cars being manufactured, brand new cars also become outdated easily and hence resale is a good option to buy a high-end car at a reasonable rate. In most countries, buying a brand new car can be expensive. Few people prefer to buy a second-hand car and would require a proper idea for selecting a vehicle that matches their budget. Approaching a broker or an agent directly wouldn't be the best thing to do as they can manipulate the customers. With the help of this project the customers can get an idea of what they should be looking for.

Car resale value prediction is a comprehensive task which involves consideration of various attributes such as kilometers run, age of the car, power, etc. In addition to these features, brand and model of the car might play a major role in prediction. Due to rising fuel prices, fuel economy is also of prime importance. Consideration of damage and repair is also required as a slightly damaged car is less likely to have a good price.

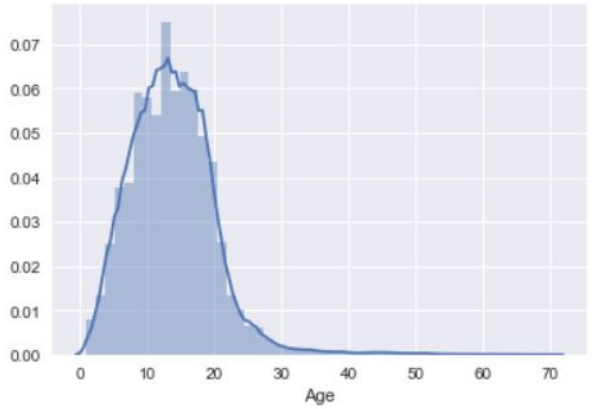
Objective

The aim of the project is to build a car resale value prediction system based on a wide range of car features by using a Neural Network approach to obtain a good amount of accuracy. The attributes used in this project are brand, model, fuel type, vehicle type, kilometers run, power, unrepaired damage, gearbox and registration date.

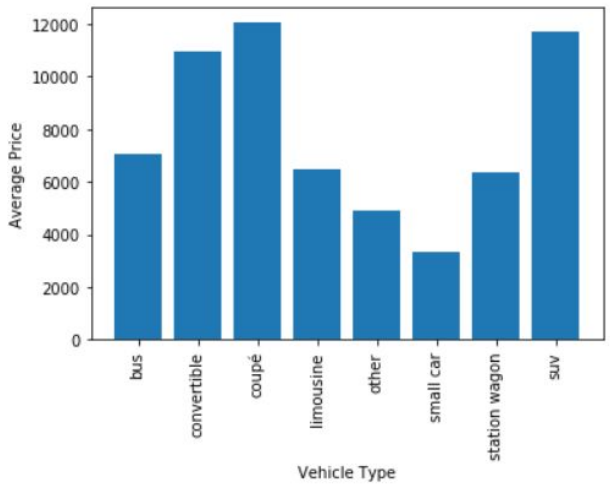
Lots of techniques for the prediction have been proposed to solve the problem. Each technique has its own advantage of solving specific problems. In this project a neural network implementation has been made which predicts the value in a robust manner. The network consists of 5 hidden layers.

Methodology, Tools, and Datasets

Methodology:

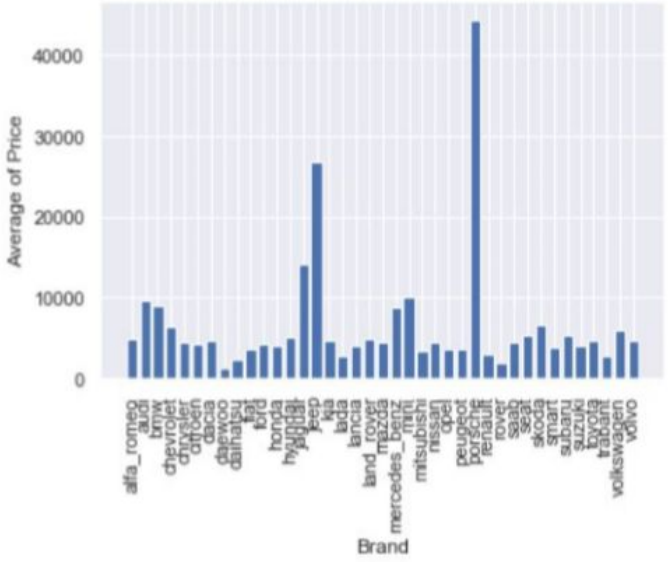


Data cleaning and visual representation of data plays an important role. Cleaning refers to sorting out the data that is incomplete. But not all data can be filled based on the previous learning, Gear type, fuel type are some of the main factors on which the classification depends. If there are only 2 options in the given type, then we can classify them as 0 and 1 values.

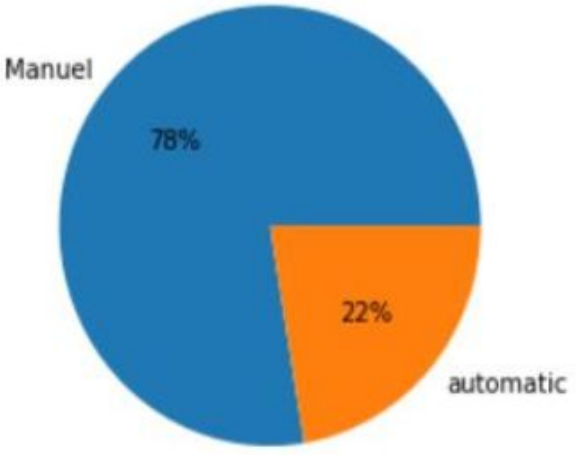


By using graphical representations like histograms, pie charts, and continuous graphs, it becomes easier to learn about the pattern of the type of cars in demand. Some general observations that can be made are that SUV and Coupe types are costlier and people prefer a manual gearbox.

Fig. 1. Price (in Dollars) vs Brand



Pie Chart showing Automatic:Manual Gearbox Ratio



Artificial Neural Network **:**

Numerous advances have been made in the field of computer science during the past two to three decades and the most notable among those are the domains of artificial intelligence (AI) and machine learning (ML). It initially started off with procedures like linear or logistic regression, k-Nearest Neighbors (KNN) among others which are mainly derived from a mathematical or statistical background, to be specific. Another such ML technique is the Artificial Neural Network (ANN). This was inspired by the way a human brain works. It is being used in several domains to solve problems such as pattern recognition, speech-to-text systems in the modern world.

Working of an ANN**:**

An ANN is made up of a set of nodes called neurons. These neurons together make up a layer of an ANN which is connected to the previous and the successive layer. Thus, the kind of layers in an ANN are three: an input layer to which the input is given, a hidden layer which takes in the inputs from the previous node and outputs the data obtained to the next layer and finally comes the output layer which simply only processes the input given to it and produces a certain output.

$$y(i) = f(i) \sum_{j=1}^{n} (w(i,j)\, x(j) - \theta(i))$$

where y is equal to the output of the neuron i, x is the input j to the node and w is the weight associated with the input j, f is called the activation function and theta is the constant (or the bias associated with the neuron i. Some of the activation functions commonly in use are sigmoid, Relu, softmax and tanh.

Before using the ANN to actually perform the desired task, it should be exposed to different possibilities the input may be given to it. By possibilities, it is implied towards the various ranges of values every attribute may possess. This process by which an ANN learns the data is called training of ANN. In this process, all the weights which are associated with a neuron are tuned so that once training is done, ANN can give the best output possible. In other words, it can be stated that the accuracy of the output which the ANN gives us is directly proportional to how well the weights are tuned over the process of training.

The training process of the ANN goes as follows. First, the input vector is passed into the input layer. All these values are passed through an activation function and all the values are then accumulated at every node in the subsequent layer.

This process continues until the output layer gives some results after passing the output value through some activation function. The training algorithm proceeds further such that some overall error value is minimized. These values are called loss functions and some loss functions which are commonly used are categorical cross-entropy, binary cross-entropy, mean squared error or mean absolute error among others. Once the output is computed, the output is propagated backward through gradients which are computed at every stage and are used in the updating of weights. The algorithm that is used to compute the gradient at every stage is called backpropagation.

Once all the training is done, the next task is to check the accuracy of the generated model by passing unseen data to the ANN for prediction. Here, the output produced by the ANN is stored somewhere so that the performance can be judged later.

Tools:

The model uses the following modules in python.

1. Pandas - To read the CSV file
2. Keras - To build the neural network model.
3. Sklearn - To evaluate the predicted values, in other words, to determine the accuracy.

Datasets:

The ‘Used Car Listing’ dataset (335000 x 16) has been chosen from Kaggle. There are a lot of attributes mentioned in the datasets but we’ll be using only brand, model, fuel type, vehicle type, kilometers run, power, unrepaired damage, gearbox, and registration date. The assumption made for this project is that the price range of cars is still valid even though the dataset is 2-3 years old. Due to inflation,the time gap might have a considerable impact on the car prices.The following table describes the Distribution of Data, and the Unique Categories present with it.

Attribute	*Unique Categories*
Model	243
Brand	39
Vehicle Type	8
Fuel Type	5
Unrepaired Damage	2
Gearbox	2

TABLE I
DISTRIBUTION OF DATA

Implementation of ANN

To solve the problem at hand, ANN was constructed using the Keras library designed for python, which internally uses TensorFlow or Theano libraries as the backend. Here we use TensorFlow as a backend.
The input layer consists of a number of neurons equal to the number of features in the input vector.
The activation functions used are ‘ReLu’ and ‘linear’.
The model consists of 6 hidden layers where 2 of those layers were used for regularization finally passing to the output layer.
Regularization is a technique which is used to overcome the overfitting problem during training.
In order to reduce the errors in prediction, we use a loss function whose main objective is to minimize its values during the flow of the training phase. The function we used here is ‘mean_absolute_error’ and the optimizer used is ‘adam’.

We have shown a summary of the neural network model used below.

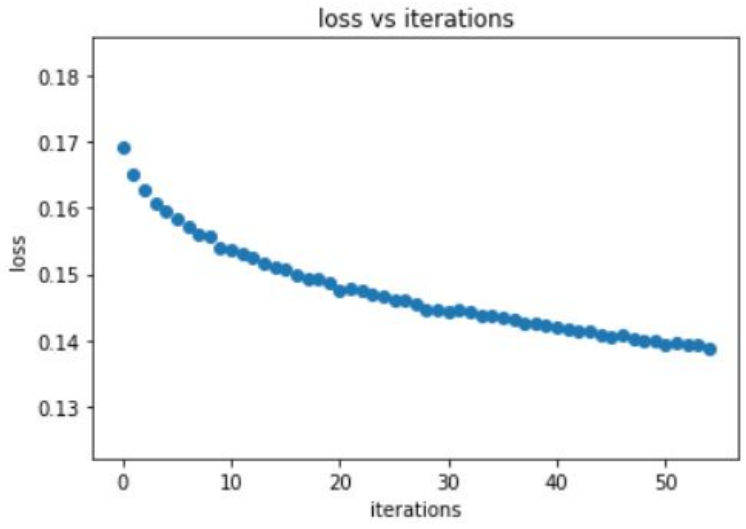
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               38016
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               65792
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 257
=================================================================
Total params: 202,881
Trainable params: 202,881
Non-trainable params: 0
```

Result and Analysis:

Validation of a model is important as it gives us an idea of how it will behave with unseen data. Based on these results, one can find out whether the model over-fits or under-fits the data, or is a generalized representation. To achieve this, a portion of the data is kept aside. The model is trained on the remaining data. The model is then evaluated on the portion of data that was left out. The data was split into various ratios. The average of the accuracies recorded serves as the performance metric for the model.

Since linear regression was used in the output layer of ANN, the accuracies were reported by R-squared values. It is the proportion of variance in the observed data that is explained by the model. The value of R square lies between 0 and 1. Higher values are better because it means that more variance is explained by the model. The following table talks about the R square value and the mean absolute error obtained for different fold values.

Folds	*R-squared value*	*Mean_absolute_error*
60-40	0.862	0.1369
70-30	0.852	0.1402
80-20	0.867	0.1418
90-10	0.851	0.1384
Average	0.858	0.1393



Comparison of loss and number of iterations

As we can see from the diagram the loss value decreases as the number of iterations increases and becomes constant after a certain number of iterations.

Conclusion and Future Scope:

Conclusion:

With the ANN model used for the prediction of the car resale value R square value of 0.858 was obtained with a mean absolute error of less than 0.15 the prediction accuracy turns out to be around 85%. This can be further improved by adding more attributes to the model and adding a few more hidden layers wherever required.

Future Scope:

As few attributes were not used from the datasets and few attributes such as interior accessories (stereo, GPS navigation, seat covers, etc.), are not present in the datasets, those factors will have to be considered to develop a model which can predict more precise values of the car. Further we can develop a way of listing vintage cars separately and predict their prices. Various boosting algorithms such as XGBoost can be considered for better prediction.

References

1. Kaggle site for dataset.
2. Cars24 (To check the factors it uses for prediction). https://www.cars24.com/
3. Journal on resale value prediction. http://www.temjournal.com/content/81/TEMJournalFebruary2019_113_118.pdf
4. TowardsDataScience website. https://towardsdatascience.com/car-selection-and-sales-day-prediction-8c4a474f9dca
5. IEEE paper: A New Model for Residual Value Prediction of the Used Car Based on BP Neural Network and Nonlinear Curve Fit https://sci-hub.tw/https://ieeexplore.ieee.org/document/5721273

Appendix

Code:

- **Model training and generation of model weights**

```
import pandas as pd
import numpy as np
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from numpy import array, reshape, mean, std
from scipy.stats import zscore
from sklearn.preprocessing import OneHotEncoder
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from sklearn.metrics import r2_score
df = pd.read_csv("cleaned_dataset.csv")
y = array(df['dollar_price'])

df = df.drop(['dollar_price','vehicle_bin', 'brand_bin', 'model_tag', 'postal_code', 'date_crawled', 'name', 'registration_year', 'gearbox', 'registration_month', 'unrepaired_damage', 'Unnamed: 0', 'last_seen_online', 'Diff_Last_Ad', 'ad_created'], axis=1)
temp = df.loc[:, ['model', 'brand', 'vehicle_type', 'fuel_type', 'damage_bin', 'gear_bin']]
df.drop(['model', 'brand', 'vehicle_type', 'fuel_type', 'damage_bin', 'gear_bin'], axis=1, inplace=True)

def myOHE(df, column, ohe_object):
    # Encode the column
    column_encoded = array(temp[column]).reshape(-1, 1)
    column_encoded = ohe.fit_transform(column_encoded)
    fff=0
    # Add the attributes to the dataframe
    for i in range(len(column_encoded[0]) - 1):
        df['{}_{}'.format(column, str(i))] = column_encoded[:, i]
        #--------------------
        if(fff==0):
            print(column_encoded[:,1])
            fff=fff+1
```

```
    #------------------

    # Drop the 'column' in the dataframe
    df.drop(column, axis=1, inplace=True)
    # Return the dataframe
    return df

ohe = OneHotEncoder(sparse=False)
temp = myOHE(temp, 'brand', ohe)
temp = myOHE(temp, 'model', ohe)
temp = myOHE(temp, 'fuel_type', ohe)
temp = myOHE(temp, 'vehicle_type', ohe)
#Creating a numpy array of the independent variables in the dataset
df = pd.DataFrame(zscore(df))
df = pd.concat([df, temp], axis=1)
X = array(df)

# Scale down the values of the cost
MEAN = mean(y)
STD = std(y)
y = zscore(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.142857,
random_state=0)

model = Sequential()
model.add(Dense(128,input_dim=(X_train[0]).shape[0],activation='relu'
,kernel_initializer='normal'))
model.add(Dense(256, activation='relu', kernel_initializer='normal'))
model.add(Dropout(0.05))
model.add(Dense(256, activation='relu', kernel_initializer='normal'))
model.add(Dropout(0.05))
model.add(Dense(256, activation='relu', kernel_initializer='normal'))
model.add(Dense(1, activation='linear', kernel_initializer='normal'))
print(model.summary())

es = EarlyStopping(monitor='val_mean_absolute_error', min_delta=0.001, patience=20)
adam = Adam(clipnorm=1.)
model.compile(loss='mean_absolute_error', optimizer=adam, metrics=['mean_absolute_error'])
model.fit(X_train, y_train, batch_size=128, epochs=50, verbose=1, validation_data=(X_val,
y_val), callbacks=[es])
predictions = model.predict(X_test)
```

```
for i in range(10):
    print(predictions[i], y_test[i]) #check the predicted values

val = len(predictions)
count = 0
print('Result ',result)
model.save("model.h5")
```

- **Prediction Code using model weights**

```
from datetime import datetime
from numpy import loadtxt
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from keras.models import load_model
import tensorflow as tf
global graph
model = load_model('model.h5')

def myOHE(df, column, ohe,mappings):
    # Encode the column
    column_encoded = np.array(df[column]).reshape(-1, 1)
    column_encoded = ohe.fit_transform(column_encoded)
    for i,j in zip(df[column],column_encoded):
        if i not in mappings:
            mappings[i]=j
    # Add the attributes to the dataframe
    for i in range(len(column_encoded[0]) - 1):
        df['{}_{}'.format(column, str(i))] = column_encoded[:, i]

    # Drop the 'column' in the dataframe
    df.drop(column, axis=1, inplace=True)
    # Return the dataframe
    return df
def Age(age):
    date1=str(datetime.today()-datetime.strptime(age,'%d-%m-%Y'))
    date2=date1.split(",")[0]
    date3=int(date2.split(" ")[0])
    return date3/365
def prediction(model1,brand,fuel,vehicle,damage,gear,power,km,age):
```

```
    df = pd.read_csv("cleaned_dataset.csv")
    y = np.array(df['dollar_price'])
    MEAN = np.mean(y)
    STD = np.std(y)
     df = df.drop(['dollar_price','vehicle_bin', 'brand_bin', 'model_tag', 'postal_code', 'date_crawled',
'name', 'registration_year', 'gearbox', 'registration_month', 'unrepaired_damage', 'Unnamed: 0',
'last_seen_online', 'Diff_Last_Ad', 'ad_created'], axis=1)
    temp = df.loc[:, ['model', 'brand', 'vehicle_type', 'fuel_type', 'damage_bin', 'gear_bin']]
          df.drop(['model', 'brand', 'vehicle_type', 'fuel_type', 'damage_bin', 'gear_bin'], axis=1,
inplace=True)
    mean_power = np.mean(df['power_ps'])
    mean_age = np.mean(df['Age'])
    mean_km = np.mean(df['kilometer'])
    std_power = np.std(df['power_ps'])
    std_age = np.std(df['Age'])
    std_km = np.std(df['kilometer'])
    z_pow = (power - mean_power)/std_power
    z_km = (km - mean_km)/std_km
    z_age = (Age(age) - mean_age)/std_age
    ohe = OneHotEncoder(sparse=False)
    mappings = {}
    temp = myOHE(temp, 'brand', ohe,mappings)
    temp = myOHE(temp, 'model', ohe,mappings)
    temp = myOHE(temp, 'fuel_type', ohe, mappings)
    temp = myOHE(temp, 'vehicle_type', ohe, mappings)
    X = []
    X.append(z_pow)
    X.append(z_km)
    X.append(z_age)
    X.append(damage)
    X.append(gear)
    X1=np.append(mappings[brand][:-1],mappings[model1][:-1])
    X2 = np.append(mappings[fuel][:-1],mappings[vehicle][:-1])
    X3 = np.append(X1,X2).tolist()
    X4 = X+X3
    value = model.predict(np.array([X4]),verbose = 0)
    val = (value[0][0]*STD)+MEAN
    return val
value = str(prediction('golf','bmw','diesel','station wagon',0,0,193,150000,'24-12-1999'))
print('value', value)
```

Sample Data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		date_craw	name	dollar_pric	vehicle_ty	registratio	gearbox	power_ps	model	kilometer	registratio	fuel_type	brand	unrepaired	ad_create	postal_co	last_seen_	model_tag	Age	Diff_Last_	vehicle_bi	brand_bin	damage_b
2	2	########	GOLF_4_1	1500	small car	2001	manuell	75	golf	150000	6	gasoline	volkswage	nein	########	91074	########	7	16.25	0 days 00:(	101	100101	0
3	3	########	Skoda_Fab	3600	small car	2008	manuell	69	fabia	90000	7	diesel	skoda	nein	########	60437	########	6	9.333333	6 days 00:(	101	11111	0
4	4	########	BMW_316	650	limousine	1995	manuell	102	3er	150000	10	gasoline	bmw	ja	########	33775	########	8	22.5	2 days 00:(	11	10	1
5	5	########	Peugeot_2	2200	convertibl	2004	manuell	109	2_reihe	150000	8	gasoline	peugeot	nein	########	67112	########	5	13.33333	4 days 00:(	1	11001	0
6	6	########	Ford_C___	14500	bus	2014	manuell	125	c_max	30000	8	gasoline	ford	nein	########	94505	########	8	3.333333	0 days 00:(	0	1010	0
7	8	########	Mazda_3_	2000	limousine	2004	manuell	105	3_reihe	150000	12	gasoline	mazda	nein	########	96224	########	6	13.75	11 days 00	11	10011	0
8	9	########	Volkswage	2799	station wa	2005	manuell	140	passat	150000	12	diesel	volkswage	ja	########	57290	########	7	12.66667	0 days 00:(	110	100101	1
9	10	########	VW_Passa	999	station wa	1995	manuell	115	passat	150000	11	gasoline	volkswage	nein	########	37269	########	7	22.66667	17 days 00	110	100101	0
10	13	########	Renault_T	1750	small car	2004	automatik	75	twingo	150000	2	gasoline	renault	nein	########	65599	########	3	12.08333	17 days 00	101	11011	0
11	14	########	Ford_C_M	7550	bus	2007	manuell	136	c_max	150000	6	diesel	ford	nein	########	88361	########	8	10.25	13 days 00	0	1010	0
12	15	########	Mercedes_	1850	bus	2004	manuell	102	a_klasse	150000	1	gasoline	mercedes_	nein	########	49565	########	6	12.25	4 days 00:(	0	10100	0
13	16	########	Volkswage	10400	coupÃ©	2009	manuell	160	scirocco	100000	4	gasoline	volkswage	nein	########	75365	########	10	7	4 days 00:(	10	100101	0
14	17	########	BMW_530	3699	limousine	2002	automatik	231	5er	150000	7	gasoline	bmw	nein	########	68309	########	9	15.33333	11 days 00	11	10	0
15	19	########	MERCEDE	500	limousine	1990	manuell	118	andere	150000	10	gasoline	mercedes_	ja	########	35390	########	9	27.58333	0 days 00:(	11	10100	1
16	20	########	BMW_530	2500	station wa	2002	automatik	193	5er	150000	9	diesel	bmw	ja	########	73765	########	9	15.5	0 days 00:(	110	10	1
17	22	########	Honda_Civ	6900	limousine	2008	manuell	99	civic	60000	11	gasoline	honda	nein	########	12621	########	5	9.666667	19 days 00	11	1011	0
18	23	########	Volkswage	1990	bus	1981	manuell	50	transporte	5000	1	gasoline	volkswage	nein	########	87471	########	10	35.16667	1 days 00:(	0	100101	0
19	24	########	Fiat_Punto	690	small car	2003	manuell	60	punto	150000	3	gasoline	fiat	nein	########	86199	########	3	13	1 days 00:(	101	1001	0
20	25	########	Mercedes_	3300	limousine	1995	automatik	113	e_klasse	150000	1	diesel	mercedes_	nein	########	53879	########	9	21.25	2 days 00:(	11	10100	0
21	27	########	BMW_325	18000	limousine	2007	automatik	218	3er	20000	5	gasoline	bmw	nein	########	39179	########	8	10.16667	13 days 00	11	10	0
22	28	########	Mercedes_	3500	limousine	2004	automatik	122	e_klasse	150000	11	diesel	mercedes_	nein	########	67071	########	9	13.66667	13 days 00	11	10100	0
23	31	########	Abschlepp	11900	other	2002	manuell	129	andere	150000	11	diesel	volkswage	nein	########	10551	########	9	15.58333	4 days 00:(	100	100101	0
24	32	########	Mercedes_	1500	bus	1984	manuell	70	andere	150000	8	diesel	mercedes_	nein	########	22767	########	9	33.41667	2 days 00:(	0	10100	0
25	34	########	BMW_E60	12500	limousine	2006	automatik	231	5er	150000	11	diesel	bmw	nein	########	46119	########	9	11.66667	12 days 00	11	10	0
26	35	########	Mini_One_	6990	limousine	2007	manuell	95	one	100000	8	gasoline	mini	nein	########	59174	########	8	10.41667	8 days 00:(	11	10101	0
27	36	########	Smart_For	3900	small car	2008	automatik	61	fortwo	80000	6	gasoline	smart	nein	########	21073	########	6	9.25	2 days 00:(	101	100000	0
28	37	########	Renault_C	590	small car	1999	manuell	75	clio	125000	8	gasoline	renault	nein	########	84180	########	4	18.41667	0 days 00:(	101	11011	0
29	38	########	BMW_120	7999	limousine	2007	manuell	177	1er	150000	8	diesel	bmw	nein	########	53604	########	10	10.41667	1 days 00:(	11	10	0