TOÁN ỨNG DỤNG VÀ XÁC SUẤT

Project 01

1.Data Wrangling

reading databases

In [2]:

```
import pandas as pd
import matplotlib.pylab as plt
dataX_train = pd.read_csv("X_train.csv")
dataY_train = pd.read_csv("Y_train.csv")
```

In [3]:

```
dataX_train.head()
```

Out[3]:

	id	manufacturer	model	transmission	color	odometer	year	engineFuel	engineType	eng
0	1	Hyundai	i40	automatic	red	48000	2014	gasoline	gasoline	
1	2	Mitsubishi	Carisma	mechanical	green	320000	2000	diesel	diesel	
2	3	Volkswagen	T5	mechanical	white	164000	2011	diesel	diesel	
3	4	Volkswagen	T4 Multivan	mechanical	blue	385672	1998	diesel	diesel	
4	5	Toyota	Camry	automatic	black	215652	2005	gasoline	gasoline	

5 rows × 23 columns

In [4]:

```
dataY_train.head()
```

Out[4]:

	id	price
0	1	15500.0
1	2	2800.0
2	3	16700.0
3	4	11000.0
4	5	6800.0

In [5]:

```
data=pd.concat([dataX_train,dataY_train['price']],axis=1)
```

In [6]:

```
data.shape
```

Out[6]:

```
(30000, 24)
```

Identify missing values

In [7]:

```
pd.isnull(data).sum()
```

Out[7]:

```
id                0
manufacturer      0
model             0
transmission      0
color             0
odometer          0
year              0
engineFuel        0
engineType        0
engineCapacity    9
bodyType          0
drivetrain        0
photos            0
feature_0         0
feature_1         0
feature_2         0
feature_3         0
feature_4         0
feature_5         0
feature_6         0
feature_7         0
feature_8         0
feature_9         0
price             0
dtype: int64
```

Dealing with missing data by droping the whole row

In [10]:

```
data=data.dropna()
data.shape
```

Out[10]:

```
(29991, 24)
```

In [11]:

```
pd.isnull(data).sum()
```

Out[11]:

```
id                0
manufacturer      0
model             0
transmission      0
color             0
odometer          0
year              0
engineFuel        0
engineType        0
engineCapacity    0
bodyType          0
drivetrain        0
photos            0
feature_0         0
feature_1         0
feature_2         0
feature_3         0
feature_4         0
feature_5         0
feature_6         0
feature_7         0
feature_8         0
feature_9         0
price             0
dtype: int64
```

Type of feature

In [13]:

```
data.dtypes
```

Out[13]:

```
id                   int64
manufacturer        object
model               object
transmission        object
color               object
odometer             int64
year                 int64
engineFuel          object
engineType          object
engineCapacity     float64
bodyType            object
drivetrain          object
photos               int64
feature_0             bool
feature_1             bool
feature_2             bool
feature_3             bool
feature_4             bool
feature_5             bool
feature_6             bool
feature_7             bool
feature_8             bool
feature_9             bool
price              float64
dtype: object
```

2.Visualization

In [14]:

```
data.corr()
```

Out[14]:

	id	odometer	year	engineCapacity	photos	feature_0	feature_1
id	1.000000	0.000493	0.000165	-0.004775	0.001453	-0.008150	0.004499
odometer	0.000493	1.000000	-0.534764	0.093426	-0.140852	0.173271	-0.211673
year	0.000165	-0.534764	1.000000	0.006046	0.253406	-0.384642	0.460675
engineCapacity	-0.004775	0.093426	0.006046	1.000000	0.114059	-0.126360	0.119021
photos	0.001453	-0.140852	0.253406	0.114059	1.000000	-0.116110	0.090568
feature_0	-0.008150	0.173271	-0.384642	-0.126360	-0.116110	1.000000	-0.668149
feature_1	0.004499	-0.211673	0.460675	0.119021	0.090568	-0.668149	1.000000
feature_2	-0.000923	-0.096220	0.217780	0.385353	0.135591	-0.281051	0.242150
feature_3	-0.001143	-0.258523	0.459394	0.247034	0.184041	-0.323355	0.312916
feature_4	-0.005315	-0.089480	0.203212	0.458804	0.142967	-0.293500	0.254367
feature_5	-0.004999	-0.272240	0.462795	0.273899	0.165465	-0.389845	0.388309
feature_6	-0.000944	-0.186163	0.370010	0.290530	0.190207	-0.237737	0.233923
feature_7	-0.000448	-0.281300	0.489654	0.204585	0.197372	-0.314294	0.308194
feature_8	-0.004446	-0.245585	0.497154	0.241620	0.201338	-0.447511	0.442187
feature_9	-0.003588	-0.132611	0.267195	0.240504	0.134835	-0.621576	0.377227
price	-0.001462	-0.415628	0.719046	0.336310	0.309603	-0.281111	0.303665

correlation of numerical variables

In [15]:

```
data[['odometer','year','engineCapacity','photos']].corr()
```

Out[15]:

	odometer	year	engineCapacity	photos
odometer	1.000000	-0.534764	0.093426	-0.140852
year	-0.534764	1.000000	0.006046	0.253406
engineCapacity	0.093426	0.006046	1.000000	0.114059
photos	-0.140852	0.253406	0.114059	1.000000

Analyzing Individual Feature Patterns using Visualization

Continuous numerical variables:

Attribute odometer

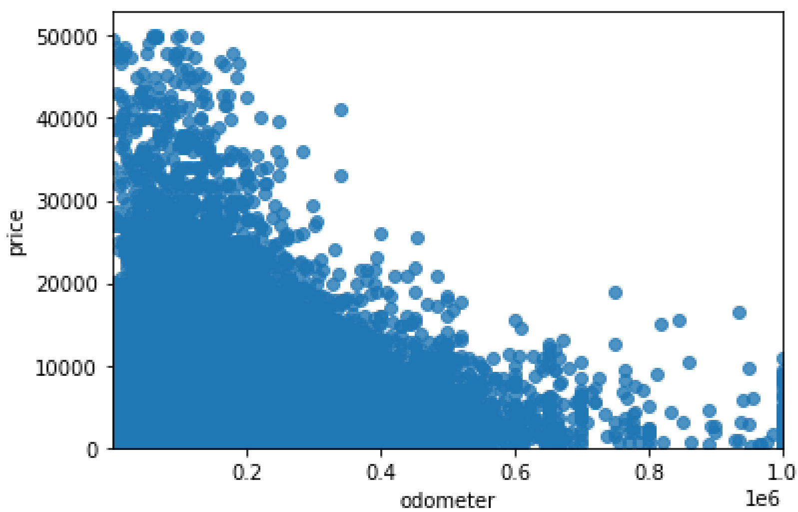
Scatterplot of "odometer" and "price"

In [17]:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.regplot(x="odometer", y="price", data=data)
plt.ylim(0,)
```

Out[17]:

```
(0.0, 52913.964331577095)
```



In [18]:

```
data[["odometer", "price"]].corr()
```

Out[18]:

	odometer	price
odometer	1.000000	-0.415628
price	-0.415628	1.000000

Attribute year

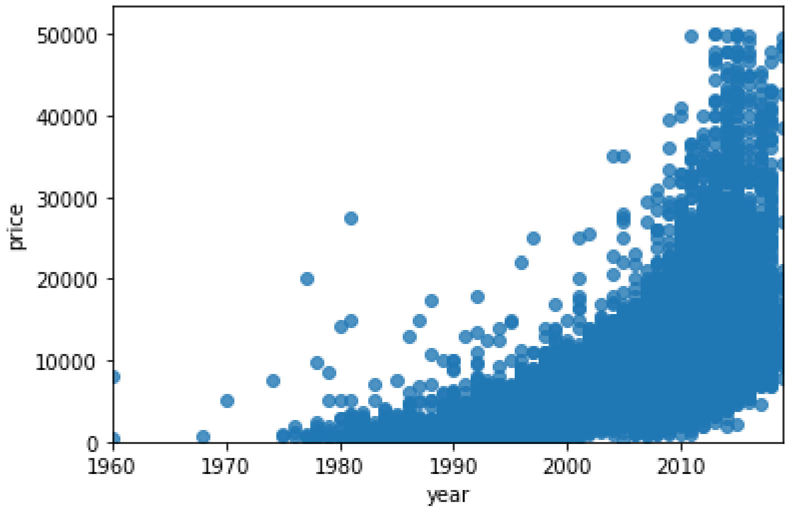
Scatterplot of "year" and "price"

In [19]:

```
sns.regplot(x="year", y="price", data=data)
plt.ylim(0,)
```

Out[19]:

```
(0.0, 53444.90652617203)
```



In [20]:

```
data[["year", "price"]].corr()
```

Out[20]:

	year	price
year	1.000000	0.719046
price	0.719046	1.000000

Attribute engineCapacity

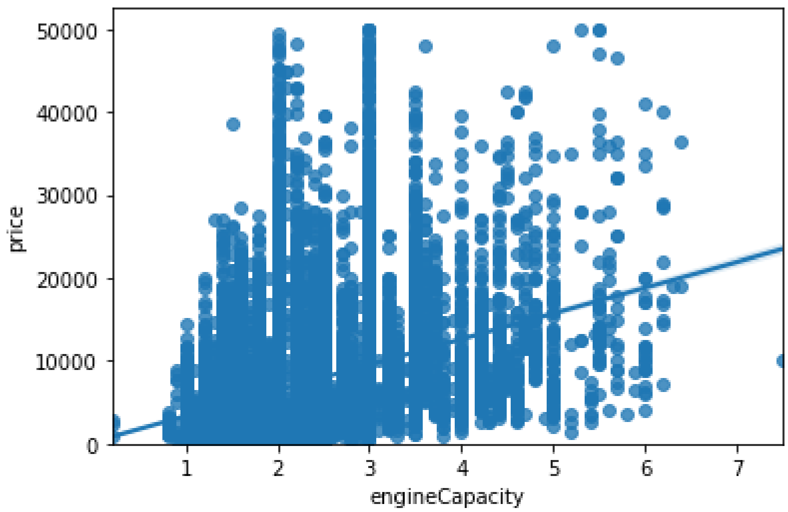
Scatterplot of "engineCapacity" and "price"

In [22]:

```
sns.regplot(x="engineCapacity", y="price", data=data)
plt.ylim(0,)
```

Out[22]:

(0.0, 52499.95)



In [23]:

```
data[["engineCapacity", "price"]].corr()
```

Out[23]:

	engineCapacity	price
engineCapacity	1.00000	0.33631
price	0.33631	1.00000

Attribute photos

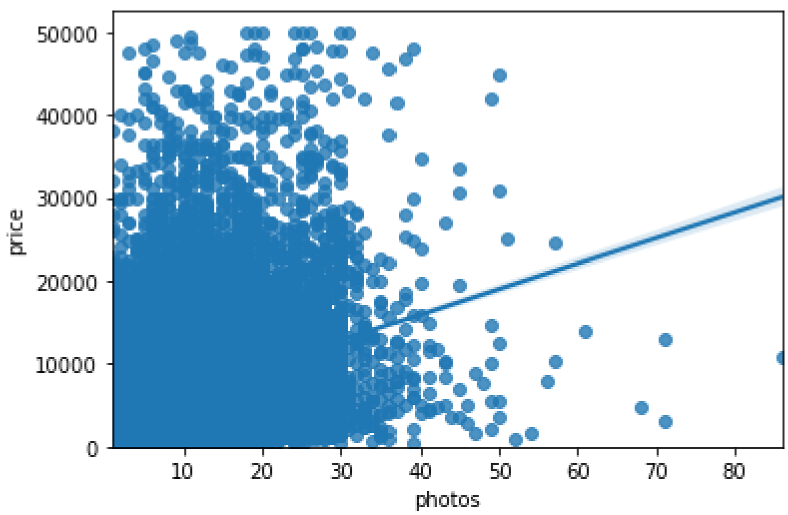
Scatterplot of "photos" and "price"

In [24]:

```
sns.regplot(x="photos", y="price", data=data)
plt.ylim(0,)
data[["photos", "price"]].corr()
```

Out[24]:

	photos	price
photos	1.000000	0.309603
price	0.309603	1.000000



Categorical variables

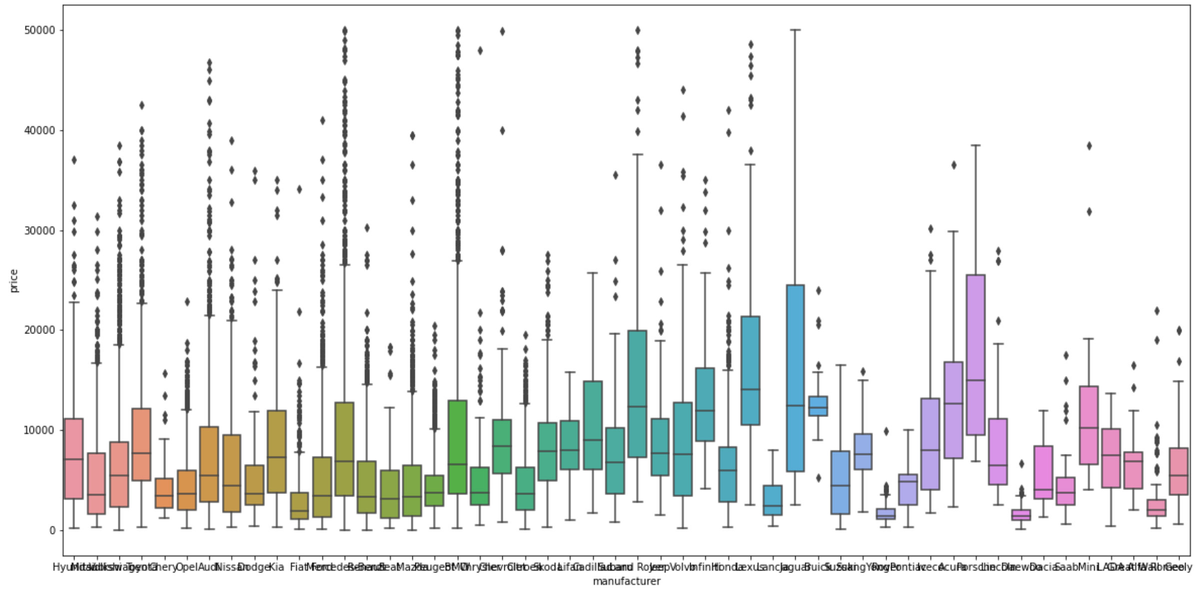
Relationship between "manufacturer" and "price"

In [26]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(x="manufacturer", y="price", data=data)
```

Out[26]:

<matplotlib.axes._subplots.AxesSubplot at 0x1fe2cc67820>



In [27]:

```
data['manufacturer'].value_counts()
```

Out[27]:

```
Volkswagen       3425
Opel             2198
BMW              2111
Ford             2064
Renault          2006
Audi             1974
Mercedes-Benz    1820
Peugeot          1562
Citroen          1267
Nissan           1095
Mazda            1071
Toyota           1028
Hyundai           893
Kia               739
Mitsubishi        710
Fiat              655
Honda             640
Skoda             609
Volvo             574
Chevrolet         350
Chrysler          276
Seat              244
Subaru            226
Dodge             202
Rover             198
Suzuki            184
Lexus             176
Alfa Romeo        171
Daewoo            163
Land Rover        153
Infiniti          129
Iveco             115
LADA              102
Saab               85
Jeep               82
Lancia             70
SsangYong          66
Acura              54
Chery              53
Geely              52
Dacia              52
Mini               51
Jaguar             47
Porsche            46
Lifan              41
Buick              39
Cadillac           34
Lincoln            30
Great Wall         30
Pontiac            29
Name: manufacturer, dtype: int64
```

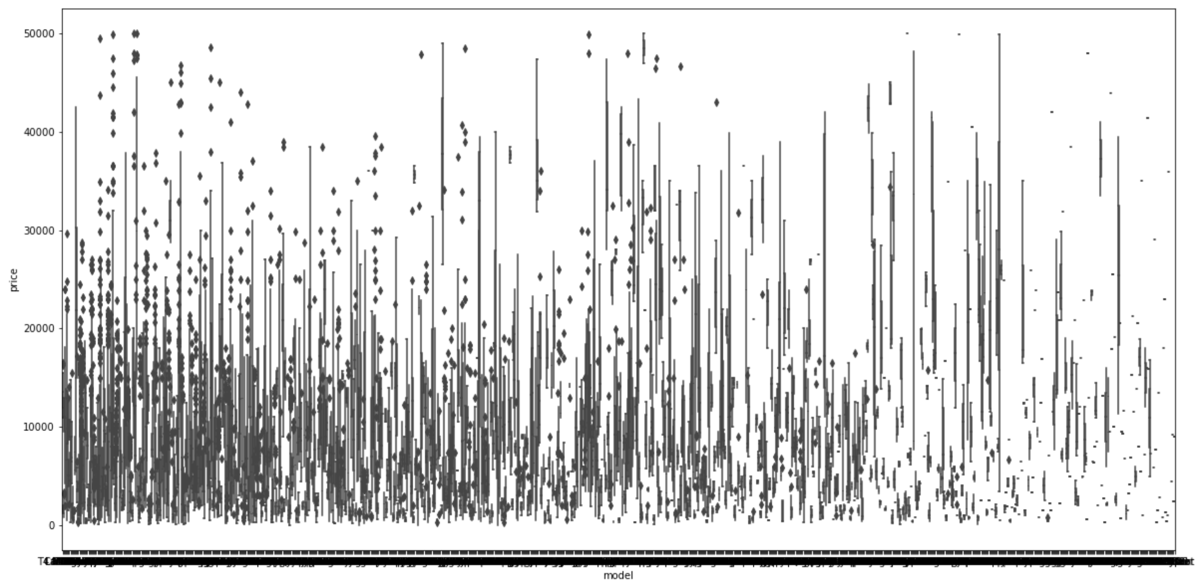
Relationship between "model" and "price"

In [29]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(x="model", y="price", data=data)
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe2b3f7a00>
```



In [30]:

```
data['model'].value_counts()
```

Out[30]:

```
Passat    1141
Astra      607
Golf       590
A6         572
Mondeo     515
          ... 
Tempo        1
Florid       1
S1000        1
235          1
Macan        1
Name: model, Length: 990, dtype: int64
```

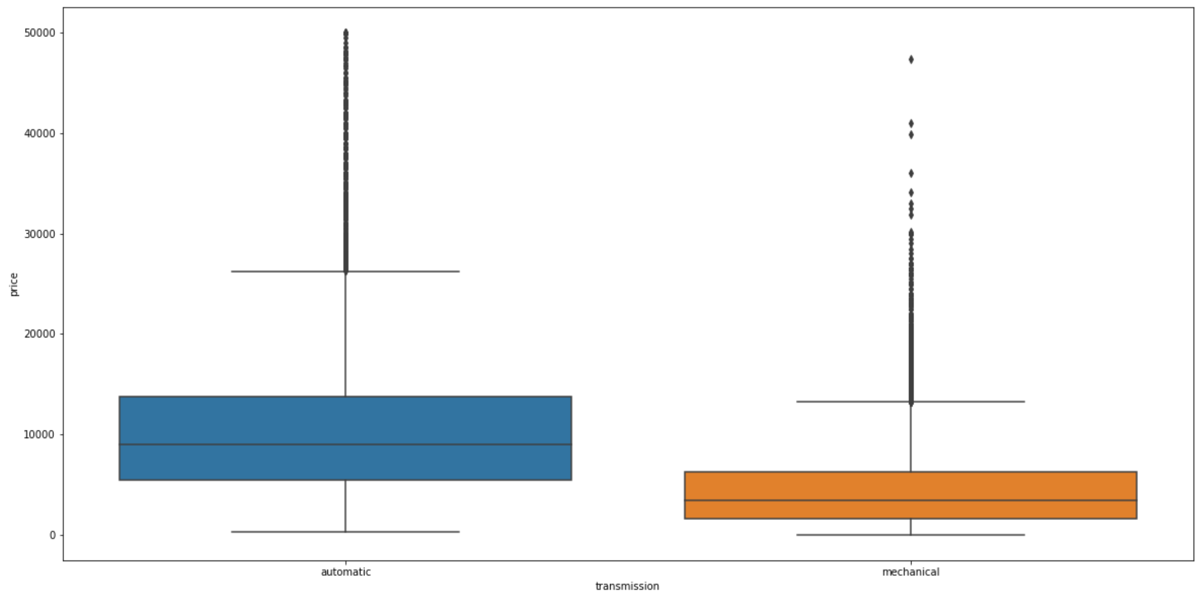
Relationship between "transmission" and "price"

In [32]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(x="transmission", y="price", data=data)
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe32f391c0>
```



In [33]:

```
data['transmission'].value_counts()
```

Out[33]:

```
mechanical    19929
automatic     10062
Name: transmission, dtype: int64
```

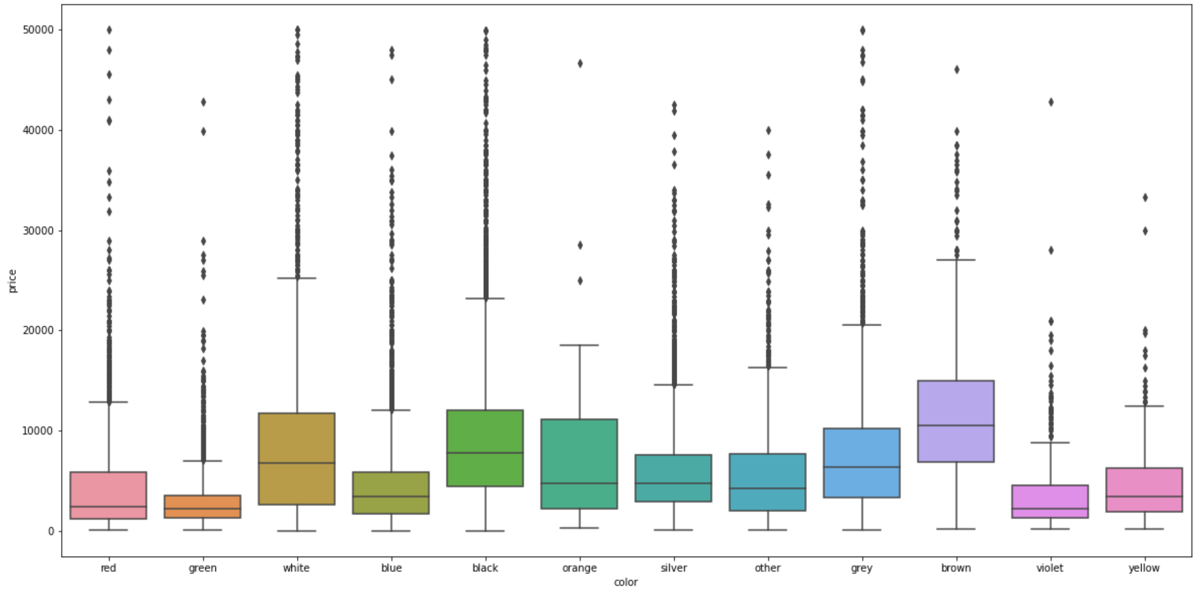
Relationship between "color" and "price"

In [34]:

```
df=data
plt.figure(figsize=(20, 10))
sns.boxplot(x="color", y="price", data=df)
```

Out[34]:

<matplotlib.axes._subplots.AxesSubplot at 0x1fe3484d940>



In [35]:

```
df['color'].value_counts()
```

Out[35]:

```
black     6115
silver    5422
blue      4509
white     3244
grey      3000
red       2266
green     2035
other     2023
brown      662
violet     371
yellow     213
orange     131
Name: color, dtype: int64
```

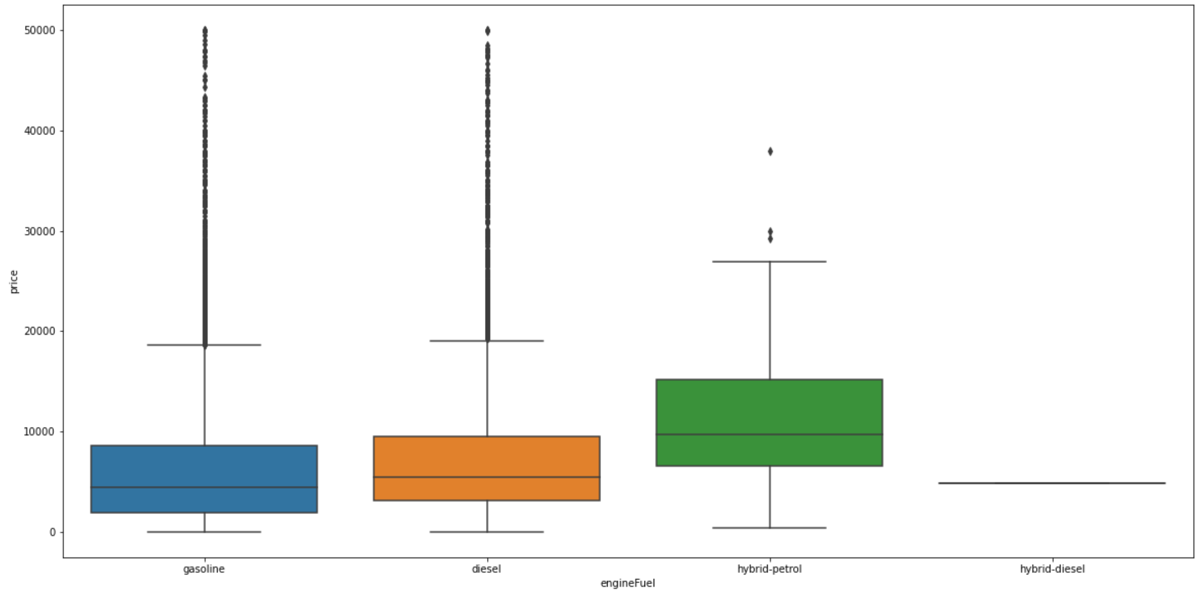
Relationship between "engineFuel" and "price"

In [36]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(x="engineFuel", y="price", data=df)
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe34767850>
```



In [37]:

```
df['engineFuel'].value_counts()
```

Out[37]:

```
gasoline         19081
diesel           10711
hybrid-petrol      198
hybrid-diesel        1
Name: engineFuel, dtype: int64
```

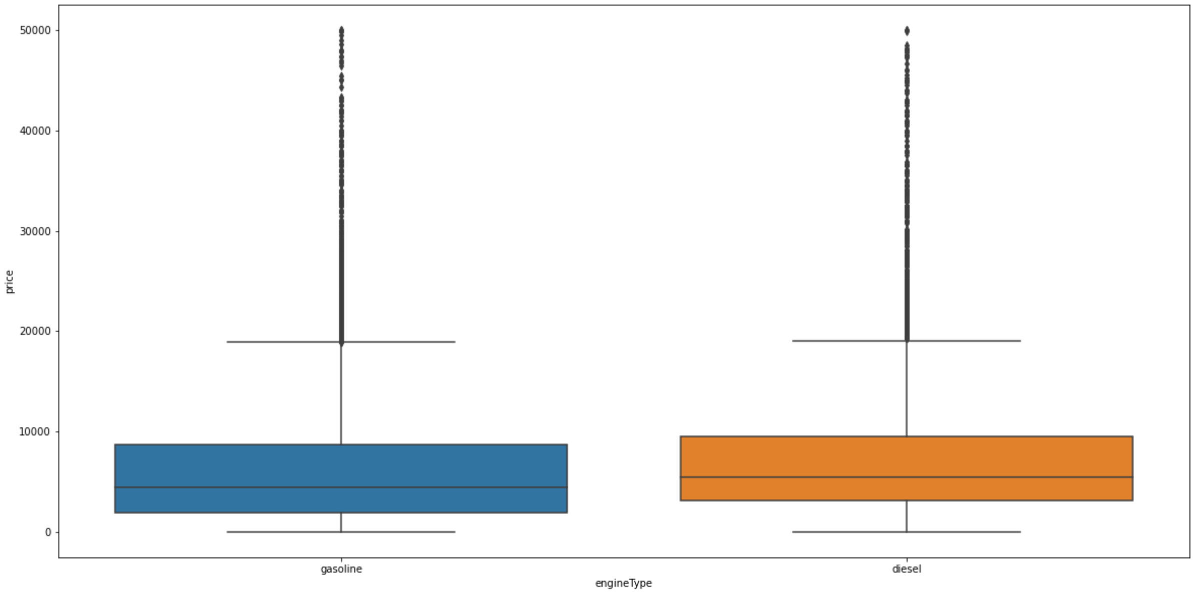
Relationship between "engineType" and "price"

In [38]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(x="engineType", y="price", data=df)
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe345cd1f0>
```



In [39]:

```
df['engineType'].value_counts()
```

Out[39]:

```
gasoline    19279
diesel      10712
Name: engineType, dtype: int64
```

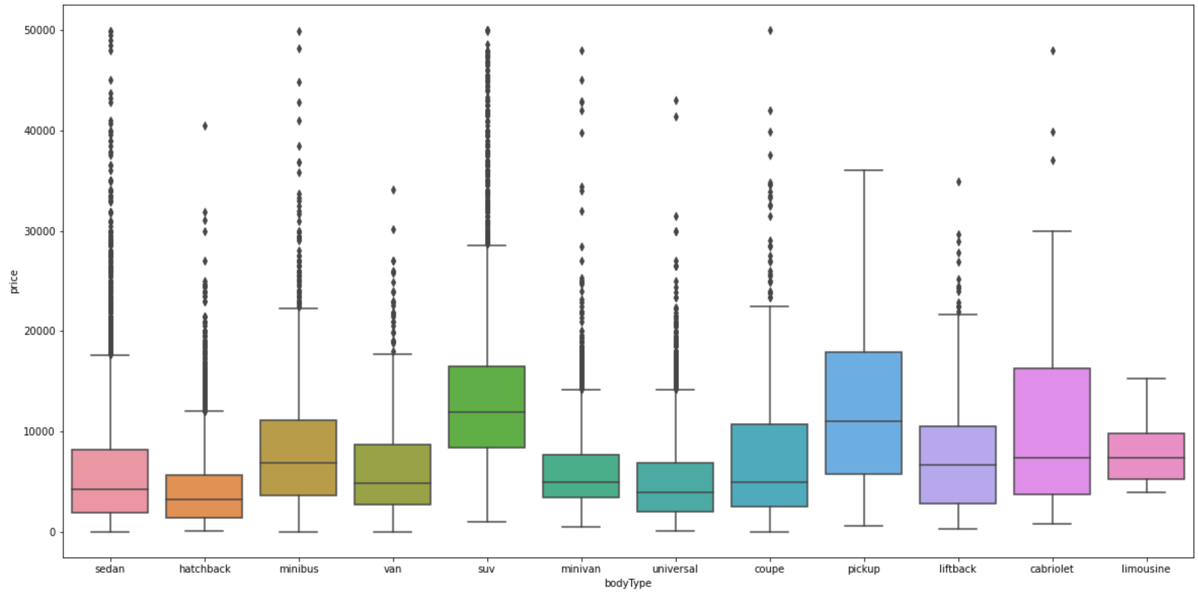
Relationship between "bodyType" and "price"

In [40]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(x="bodyType", y="price", data=df)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe3463a8b0>
```



In [41]:

```
df['bodyType'].value_counts()
```

Out[41]:

```
sedan        9897
hatchback    6128
universal    4436
suv          3868
minivan      2807
minibus      1086
van           638
coupe         518
liftback      448
pickup         91
cabriolet      63
limousine      11
Name: bodyType, dtype: int64
```

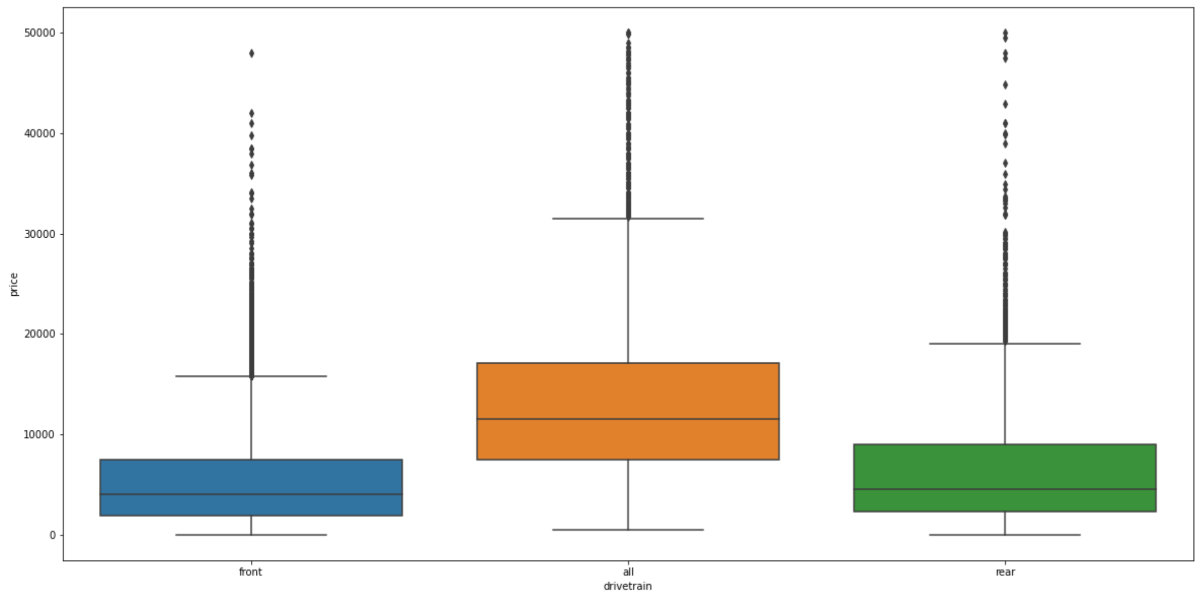
Relationship between "drivetrain" and "price"

In [42]:

```
plt.figure(figsize=(20, 10))
sns.boxplot(x="drivetrain", y="price", data=df)
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe34e201f0>
```



In [43]:

```
df['drivetrain'].value_counts()
```

Out[43]:

```
front    21928
all       4037
rear      4026
Name: drivetrain, dtype: int64
```

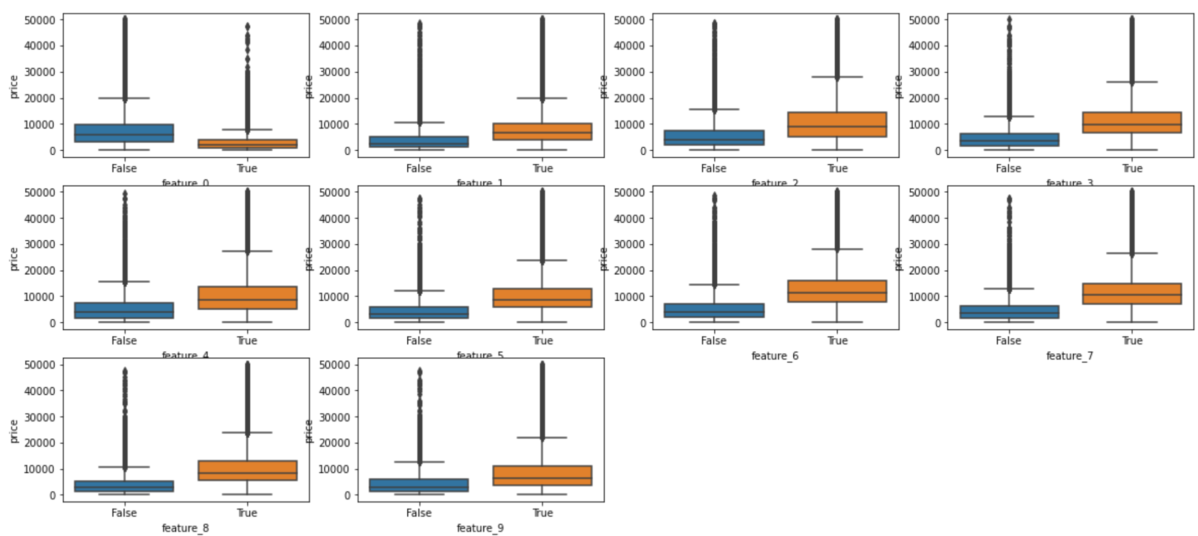
Relationship between "feature 0 1 2 3 4 5 6 7 8 9" and "price"

In [44]:

```
plt.figure(figsize = (20,12))
plt.subplot(4,4,1)
sns.boxplot(x = 'feature_0', y = 'price', data = df)
plt.subplot(4,4,2)
sns.boxplot(x = 'feature_1', y = 'price', data = df)
plt.subplot(4,4,3)
sns.boxplot(x = 'feature_2', y = 'price', data = df)
plt.subplot(4,4,4)
sns.boxplot(x = 'feature_3', y = 'price', data =df)
plt.subplot(4,4,5)
sns.boxplot(x = 'feature_4', y = 'price', data = df)
plt.subplot(4,4,6)
sns.boxplot(x = 'feature_5', y = 'price', data = df)
plt.subplot(4,4,7)
sns.boxplot(x = 'feature_6', y = 'price', data = df)
plt.subplot(4,4,8)
sns.boxplot(x = 'feature_7', y = 'price', data = df)
plt.subplot(4,4,9)
sns.boxplot(x = 'feature_8', y = 'price', data = df)
plt.subplot(4,4,10)
sns.boxplot(x = 'feature_9', y = 'price', data = df)
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe35f176a0>
```



In [45]:

```
df['feature_0'].value_counts()
```

Out[45]:

```
False    23756
True      6235
Name: feature_0, dtype: int64
```

In [46]:

```
df['feature_1'].value_counts()
```

Out[46]:

```
True     18887
False    11104
Name: feature_1, dtype: int64
```

In [47]:

```
df['feature_2'].value_counts()
```

Out[47]:

```
False    23053
True      6938
Name: feature_2, dtype: int64
```

In [48]:

```
df['feature_3'].value_counts()
```

Out[48]:

```
False    21447
True      8544
Name: feature_3, dtype: int64
```

In [49]:

```
df['feature_4'].value_counts()
```

Out[49]:

```
False    22580
True      7411
Name: feature_4, dtype: int64
```

In [50]:

```
df['feature_5'].value_counts()
```

Out[50]:

```
False    18993
True     10998
Name: feature_5, dtype: int64
```

In [55]:

```
df['feature_6'].value_counts()
```

Out[55]:

```
False    24677
True      5314
Name: feature_6, dtype: int64
```

In [56]:

```
df['feature_7'].value_counts()
```

Out[56]:

```
False    21790
True      8201
Name: feature_7, dtype: int64
```

In [57]:

```
df['feature_8'].value_counts()
```

Out[57]:

```
False    17011
True     12980
Name: feature_8, dtype: int64
```

In [58]:

```
df['feature_9'].value_counts()
```

Out[58]:

```
True     17859
False    12132
Name: feature_9, dtype: int64
```

Descriptive Statistical Analysis

In [59]:

```
df.describe()
```

Out[59]:

	id	odometer	year	engineCapacity	photos	price
count	29991.000000	29991.000000	29991.000000	29991.000000	29991.000000	29991.000000
mean	15000.157514	252907.284119	2003.124371	2.054022	9.701744	6596.436659
std	8660.116848	131377.237398	7.514463	0.662445	6.128716	6092.176086
min	1.000000	1.000000	1960.000000	0.200000	1.000000	1.000000
25%	7500.500000	163000.000000	1998.000000	1.600000	5.000000	2300.000000
50%	15001.000000	250000.000000	2003.000000	2.000000	8.000000	4900.000000
75%	22499.500000	326500.000000	2009.000000	2.300000	12.000000	8990.000000
max	30000.000000	1000000.000000	2019.000000	7.500000	86.000000	50000.000000

In [60]:

```
df.describe(include=['object'])
```

Out[60]:

	manufacturer	model	transmission	color	engineFuel	engineType	bodyType	drivetrain
count	29991	29991	29991	29991	29991	29991	29991	29991
unique	50	990	2	12	4	2	12	3
top	Volkswagen	Passat	mechanical	black	gasoline	gasoline	sedan	front
freq	3425	1141	19929	6115	19081	19279	9897	21928

3.Conclusion: Important Variables</p>

```
We now have a better idea of what our data looks like and which variables are imp
ortant to take into account when predicting the car price. We have narrowed it do
wn to the following variables:

Continuous numerical variables:

                    odometer
                    year
Categorical variables:
                    manufacturer
                    transmission
                    color
                    bodyType
                    drivetrain
                    feature_0
                    feature_1
                    feature_2
                    feature_3
                    feature_4
                    feature_5
                    feature_6
                    feature_7
                    feature_8
                    feature_9
```

4.TRAIN AND TEST

Dummy Categorical variables

In [62]:

```
DummyFeature=df[['feature_0', 'feature_1','feature_2','feature_3','feature_4','feature_5',
'feature_6','feature_7','feature_8','feature_9']].astype(int)
DummyManufacturer=pd.get_dummies(df['manufacturer'])
DummyModel=pd.get_dummies(df['model'])
DummyTransmission=pd.get_dummies(df['transmission'])
DummyColor=pd.get_dummies(df['color'])
DummyBodyType=pd.get_dummies(df['bodyType'])
DummyDrivetrain=pd.get_dummies(df['drivetrain'])
DummyEngineType=pd.get_dummies(df['engineType'])
DummyEngineFuel=pd.get_dummies(df['engineFuel'])
```

Simple Model for comparesion of other model

In [63]:

```
dataX=pd.concat([DummyManufacturer,DummyModel,DummyTransmission,DummyColor,df['odometer'],
df['year'],DummyEngineFuel,DummyEngineType,df['engineCapacity'],DummyBodyType,DummyDrivetr
ain,df['photos'],DummyFeature],axis=1)
dataY=df['price']
```

In [64]:

```
import sklearn.model_selection as model_selection

X_train, X_test, y_train, y_test = model_selection.train_test_split(dataX,dataY, train_siz
e=0.8,test_size=0.2, random_state=100)
```

In [65]:

```
import numpy as np
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(X_train, y_train)
y_predict= model.predict(X_test)
import math
A=(y_predict-y_test)**2
RMSE=math.sqrt(A.sum()/y_test.size);
print(RMSE)
```

```
563611274.4155275
```

MODEL_00 linear regression with Important Variables

In [67]:

```
dataX=pd.concat([df['odometer'],df['year'],DummyManufacturer,DummyTransmission,DummyColor,
DummyBodyType,DummyDrivetrain,DummyFeature],axis=1)
dataY=df['price']
```

In [68]:

```
import sklearn.model_selection as model_selection

X_train, X_test, y_train, y_test = model_selection.train_test_split(dataX,dataY, train_siz
e=0.8,test_size=0.2, random_state=100)



import numpy as np
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(X_train, y_train)
```

In [69]:

```
y_predict= model.predict(X_test)
import math
A=(y_predict-y_test)**2
RMSE=math.sqrt(A.sum()/y_test.size);
print(RMSE)
```

```
327395674.97051287
```

FINDING THE BEST MODEL

MODEL_01

Continuous numerical variables:

odometer => quadratic harm equation

year => quadratic harm equation

Categorical variables:

```
manufacturer
transmission
color
bodyType
drivetrain
feature_0
feature_1
feature_2
feature_3
feature_4
feature_5
feature_6
feature_7
feature_8
```

In [70]:

```
dataX=pd.concat([df['odometer'],df['odometer']**2,df['year'],df['year']**2,DummyManufactur
er,DummyTransmission,DummyColor,DummyBodyType,DummyDrivetrain,DummyFeature],axis=1)
dataY=df['price']
import math

import sklearn.model_selection as model_selection

X_train, X_test, y_train, y_test = model_selection.train_test_split(dataX,dataY, train_siz
e=0.8,test_size=0.2, random_state=100)

import numpy as np
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(X_train, y_train)

y_predicted = model.predict(X_test)
A=(y_predicted-y_test)**2
RMSE=math.sqrt(A.sum()/y_test.size)
print(RMSE)
```

```
2742.013157337794
```