In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
dataset=pd.read_csv(r"C:\Users\kotha\Downloads\autos.csv",encoding='latin-1',error_bad_line
```

In [3]:

```
dataset
```

Out[3]:

	dateCrawled	name	seller	offerType	price	a
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	
...	...	...	...	...	...	
371523	2016-03-14 17:48:27	Suche_t4___vito_ab_6_sitze	privat	Angebot	2200	
371524	2016-03-05 19:56:21	Smart_smart_leistungssteigerung_100ps	privat	Angebot	1199	
371525	2016-03-19 18:57:12	Volkswagen_Multivan_T4_TDI_7DC_UY2	privat	Angebot	9200	
371526	2016-03-20 19:41:08	VW_Golf_Kombi_1_9l_TDI	privat	Angebot	3400	
371527	2016-03-07 19:39:19	BMW_M135i_vollausgestattet_NP_52.720____Euro	privat	Angebot	28990	c

371528 rows × 20 columns

In [5]:

```
dataset.isnull().sum()
```

Out[5]:

```
dateCrawled                0
name                       0
seller                     0
offerType                  0
price                      0
abtest                     0
vehicleType            37869
yearOfRegistration         0
gearbox                20209
powerPS                    0
model                  20484
kilometer                  0
monthOfRegistration        0
fuelType               33386
brand                      0
notRepairedDamage      72060
dateCreated                0
nrOfPictures               0
postalCode                 0
lastSeen                   0
dtype: int64
```

In [6]:

```
dataset.drop(['seller','nrOfPictures','offerType','name','dateCrawled','dateCreated','lastS
dataset
```

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfR
0	480	test	NaN	1993	manuell	0	golf	150000	
1	18300	test	coupe	2011	manuell	190	NaN	125000	
2	9800	test	suv	2004	automatik	163	grand	125000	
3	1500	test	kleinwagen	2001	manuell	75	golf	150000	
4	3600	test	kleinwagen	2008	manuell	69	fabia	90000	
...	...	...	...	...	...	...	...	...	
371523	2200	test	NaN	2005	NaN	0	NaN	20000	
371524	1199	test	cabrio	2000	automatik	101	fortwo	125000	
371525	9200	test	bus	1996	manuell	102	transporter	150000	
371526	3400	test	kombi	2002	manuell	100	golf	150000	
371527	28990	control	limousine	2013	manuell	320	m_reihe	50000	

In [7]:

```
dataset.isnull().sum()
```

Out[7]:

```
price                   0
abtest                  0
vehicleType         37869
yearOfRegistration      0
gearbox             20209
powerPS                 0
model               20484
kilometer               0
monthOfRegistration     0
fuelType            33386
brand                   0
notRepairedDamage   72060
dtype: int64
```

In [9]:

```
a=dataset['notRepairedDamage'].mode().iloc[0]
a
```

Out[9]:

```
'nein'
```

In [16]:

```
dataset['gearbox'].fillna((dataset['gearbox'].mode().iloc[0]),inplace=True)
dataset['vehicleType'].fillna((dataset['vehicleType'].mode().iloc[0]),inplace=True)
dataset['model'].fillna((dataset['model'].mode().iloc[0]),inplace=True)
dataset['fuelType'].fillna((dataset['fuelType'].mode().iloc[0]),inplace=True)
dataset['notRepairedDamage'].fillna((dataset['notRepairedDamage'].mode().iloc[0]),inplace=T
```

In [17]:

```
dataset.isnull().any()
```

Out[17]:

```
price               False
abtest              False
vehicleType         False
yearOfRegistration  False
gearbox             False
powerPS             False
model               False
kilometer           False
monthOfRegistration False
fuelType            False
brand               False
notRepairedDamage   False
dtype: bool
```

In [19]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset["vehicleType"] =le.fit_transform(dataset["vehicleType"])
dataset["fuelType"] =le.fit_transform(dataset["fuelType"])
dataset["gearbox"] =le.fit_transform(dataset["gearbox"])
dataset["notRepairedDamage"] =le.fit_transform(dataset["notRepairedDamage"])
dataset["brand"] =le.fit_transform(dataset["brand"])
dataset["model"] =le.fit_transform(dataset["model"])
dataset["abtest"] =le.fit_transform(dataset["abtest"])
```

In [20]:

```
dataset.head()
```

Out[20]:

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthO
0	480	1	6	1993	1	0	118	150000	
1	18300	1	3	2011	1	190	118	125000	
2	9800	1	7	2004	0	163	119	125000	
3	1500	1	4	2001	1	75	118	150000	
4	3600	1	4	2008	1	69	103	90000	

In [33]:

```
x=dataset.iloc[:,1:12].values
y=dataset.iloc[:,0:1].values
```

In [34]:

```
x.shape
```

Out[34]:

```
(371528, 11)
```

In [35]:

```
from sklearn.preprocessing import OneHotEncoder
one=OneHotEncoder()
a=one.fit_transform(x[:,2:3]).toarray()
b=one.fit_transform(x[:,9:10]).toarray()
c=one.fit_transform(x[:,10:11]).toarray()
x=np.delete(x,[2,9,10],axis=1)
x=np.concatenate((a,b,c,x),axis=1)
```

In [36]:

```
x.shape
```

Out[36]:

```
(371528, 205)
```

In [37]:

```
dataset
```

Out[37]:

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	m
0	480	1	6	1993	1	0	118	150000	
1	18300	1	3	2011	1	190	118	125000	
2	9800	1	7	2004	0	163	119	125000	
3	1500	1	4	2001	1	75	118	150000	
4	3600	1	4	2008	1	69	103	90000	
...	...	...	...	...	...	...	...	...	
371523	2200	1	6	2005	1	0	118	20000	
371524	1199	1	2	2000	0	101	108	125000	
371525	9200	1	1	1996	1	102	225	150000	
371526	3400	1	5	2002	1	100	118	150000	
371527	28990	0	6	2013	1	320	148	50000	

371528 rows × 12 columns

In [59]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [109]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [110]:

```
x_train
```

Out[110]:

```
array([[-1.03766673e-02, -1.83425729e-03, -1.83425729e-03, ...,
         6.07813304e-01, -4.66418408e-01, -6.25541200e-01],
       [-1.03766673e-02, -1.83425729e-03, -1.83425729e-03, ...,
        -2.13949476e+00,  3.41232695e-01, -6.25541200e-01],
       [-1.03766673e-02, -1.83425729e-03, -1.83425729e-03, ...,
        -1.65748931e-02, -7.35635443e-01, -6.25541200e-01],
       ...,
       [-1.03766673e-02, -1.83425729e-03, -1.83425729e-03, ...,
         6.07813304e-01,  3.41232695e-01,  1.27520772e+00],
       [-1.03766673e-02, -1.83425729e-03, -1.83425729e-03, ...,
         6.07813304e-01,  1.14888380e+00, -6.25541200e-01],
       [-1.03766673e-02, -1.83425729e-03, -1.83425729e-03, ...,
         6.07813304e-01, -7.35635443e-01,  1.27520772e+00]])
```

In [111]:

```
y_train
```

Out[111]:

```
array([[    0],
       [16500],
       [  999],
       ...,
       [10400],
       [  699],
       [17400]], dtype=int64)
```

In [112]:

```
from sklearn.linear_model import LinearRegression
mlr= LinearRegression()
mlr.fit(x_train,y_train)
```

Out[112]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [113]:

```
y_pred= mlr.predict(x_test)
```

In [114]:

```
y_pred
```

Out[114]:

```
array([[1.66154480e+16],
       [1.67127540e+16],
       [1.67475735e+16],
       ...,
       [1.66679041e+16],
       [1.67222977e+16],
       [1.64944831e+16]])
```

In [115]:

```
from sklearn.metrics import r2_score
accuracy = r2_score(y_test,y_pred)
```

In [116]:

```
accuracy
```

Out[116]:

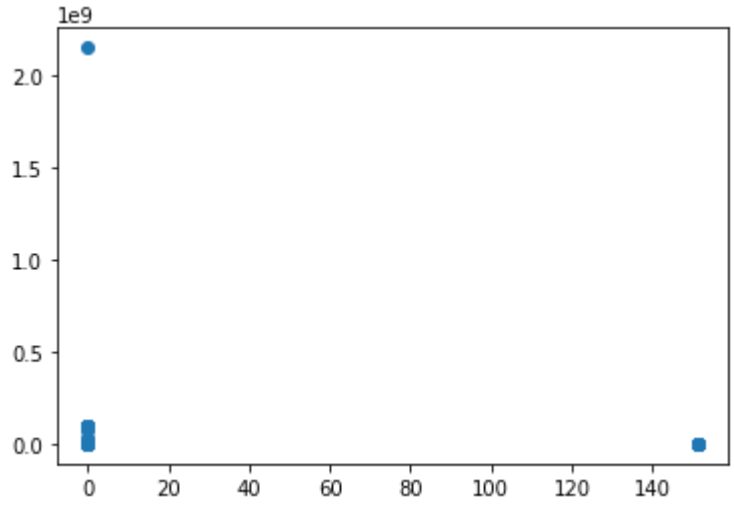
```
-6.155584490357182e+25
```

In [117]:

```
plt.scatter(x_train[:,49],y_train)
```

Out[117]:

```
<matplotlib.collections.PathCollection at 0x1ec1db4f308>
```



In [118]:

```
from sklearn.tree import DecisionTreeRegressor
decisiontree = DecisionTreeRegressor(random_state = 0)
decisiontree.fit(x_train,y_train)
```

Out[118]:

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=0, splitter='best')
```

In [119]:

```
x_test
```

Out[119]:

```
array([[-0.00898631,  1.        , -1.        , ...,  0.60790661,
         0.60953086, -0.62158458],
       [-0.00898631,  1.        , -1.        , ..., -1.61966089,
        -1.01028961,  1.28164041],
       [-0.00898631,  1.        , -1.        , ...,  0.60790661,
        -1.55022977, -0.62158458],
       ...,
       [-0.00898631,  1.        , -1.        , ...,  0.60790661,
         1.4194411 , -0.62158458],
       [-0.00898631,  1.        , -1.        , ...,  0.60790661,
        -0.47034945, -0.62158458],
       [-0.00898631,  1.        , -1.        , ...,  0.60790661,
        -1.55022977,  1.28164041]])
```

In [120]:

```
dpred = decisiontree.predict(x_test)
```

In [121]:

```
dpred
```

Out[121]:

```
array([ 3300.        , 12800.        ,   833.5       , ...,
         599.        ,  2663.33333333,  2950.        ])
```

In [122]:

```
y_test
```

Out[122]:

```
array([[ 2850],
       [11990],
       [ 7000],
       ...,
       [  750],
       [ 1650],
       [  745]], dtype=int64)
```

In [123]:

```
daccuracy= r2_score(y_test,dpred)
```

In [124]:

```
x_train.shape
```

Out[124]:

```
(297222, 205)
```

In [125]:

```
daccuracy
```

Out[125]:

```
-0.9128339111387587
```

In [126]:

```
from sklearn.ensemble import RandomForestRegressor
randomforest = RandomForestRegressor(n_estimators = 10 ,random_state = 0)
randomforest.fit(x_train,y_train)
```

```
C:\Users\kotha\anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConv
ersionWarning: A column-vector y was passed when a 1d array was expected. Pl
ease change the shape of y to (n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports
until
```

Out[126]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=No
ne,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=10, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

In [127]:

```
rpred = randomforest.predict(x_test)
```

In [128]:

```
rpred
```

Out[128]:

```
array([ 2956.        , 12381.6       ,   712.59442641, ...,
         721.65818182,  2488.9       ,  1969.9       ])
```

In [129]:

```
raccuracy = r2_score(y_test,rpred)
```

In [130]:

```
raccuracy
```

Out[130]:

```
-0.5992767421406604
```

```
#RandomForest is the best algorithm for the given dataset
```

In [ ]:

```

```