STANDARDIZATION

```
#importing standardscalar from scikitlearn to standardize data values
into standard format
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
scaled=pd.DataFrame(sc.fit_transform(Independent),columns=Independent.
columns)

scaled.head()
```

```
     seller    abtest  vehicleType  yearOfRegistration   powerPS  
kilometer  \
0 -0.002842 -0.963931    -0.914283           -1.683521 -1.712930   
0.659092   
1 -0.002842 -0.963931     1.896897            1.113393  1.237690  -
0.846937   
2 -0.002842 -0.963931     2.459133            0.025704  0.818391  -
0.846937   
3 -0.002842 -0.963931    -0.352047           -0.440448 -0.548212   
0.659092   
4 -0.002842 -0.963931    -0.352047            0.647241 -0.641389  -
2.955376   

   monthOfRegistration  fuelType  notRepairedDamage  nrOfPictures  
postalCode  \
0            -1.544670 -0.616646          -0.328996           0.0   
0.760274   
1            -0.197835  1.246101           3.039553           0.0   
0.625346   
2             0.610266  1.246101          -0.328996           0.0   
1.537240   
3             0.071532 -0.616646          -0.328996           0.0   
1.560264   
4             0.340899  1.246101          -0.328996           0.0   
0.372740   

   offerType_Gesuch  gearbox_manuell  
0         -0.005683         0.511747  
1         -0.005683         0.511747  
2         -0.005683        -1.954090  
3         -0.005683         0.511747  
4         -0.005683         0.511747  
```

DIVIDING DATA INTO TRAIN AND TEST

```
#divivng the dataset into train and test using train_test_split
function
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(scaled,Dependent,test_s
ize=0.25,random_state=0)
```

```
#shape of the training data
x_train.shape
```

```
(278646, 13)
```

```
#shape of the test data
x_test.shape
```

```
(92882, 13)
```

```
#Independent features of  training data after dividing training and
testing
x_train.head()
```

```
          seller    abtest  vehicleType  yearOfRegistration   powerPS  
\
43344  -0.002842 -0.963931    -0.352047            0.025704 -0.719037

253492 -0.002842 -0.963931     0.772425            0.647241 -0.004677

243201 -0.002842  1.037418     0.210189            0.647241  0.228267

317331 -0.002842 -0.963931    -0.914283            0.181089 -1.712930

356702 -0.002842  1.037418     1.896897            1.734930  2.480056


        kilometer  monthOfRegistration  fuelType  notRepairedDamage  \
43344    0.659092            -0.467202  1.246101           3.039553
253492  -0.846937            -1.005936  1.246101          -0.328996
243201  -0.846937             1.418367 -0.616646          -0.328996
317331   0.659092            -1.544670  1.246101          -0.328996
356702  -0.809666             0.610266 -0.616646          -0.328996

        nrOfPictures  postalCode  offerType_Gesuch  gearbox_manuell
43344            0.0    0.009471         -0.005683         0.511747
253492           0.0   -1.049561         -0.005683         0.511747
243201           0.0    0.640540         -0.005683         0.511747
317331           0.0    0.334056         -0.005683         0.511747
356702           0.0    1.162149         -0.005683        -1.954090
```

```
#Dependent feature of tetsing data after dividing training and testing

y_train.head()
```

```
43344      1500.0
253492     3500.0
243201     6990.0
317331     2500.0
356702    16275.0
Name: price, dtype: float64
```

MODEL I : RANDOMFOREST REGRESSOR

```
from sklearn.ensemble import RandomForestRegressor

#training the data to randomforestregression algorithm
rfr=RandomForestRegressor()
model=rfr.fit(x_train,y_train)

#predicting the test data
y_pred=model.predict(x_test)
```

After checking all the algorithms like Linear Regression,Decision Tree Regression,Lasso Regression,Ridge Regression and RandomForest Regression etc., ***The accuracy of the Random Forest Algorithm is high. So RandomForestRegression is the best algorithm***