generate_new_features

January 12, 2019

In [4]:
```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
cali = datasets.california_housing.fetch_california_housing()

X = cali['data']
Y = cali['target']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.8)
```

```
d:\python\lib\site-packages\sklearn\model_selection\_split.py:2179: FutureWarning: From versio
  FutureWarning)
```

In [5]:
```
# MAE = Mean Absolute Error
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor()
regressor.fit(X_train, Y_train)
Y_est = regressor.predict(X_test)

print('MAE=', mean_squared_error(Y_test, Y_est))
```

```
MAE= 1.0822040250181173
```

In [7]:
```
# Standaryzation, StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

regressor = KNeighborsRegressor()
regressor.fit(X_train_scaled, Y_train)
Y_est = regressor.predict(X_test_scaled)

print('MAE=', mean_squared_error(Y_test, Y_est))
```

```
MAE= 0.42766978453844084
```

```
In [9]: # Standaryzation, RobustScaler
        from sklearn.preprocessing import RobustScaler
        scaler = RobustScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

        regressor = KNeighborsRegressor()
        regressor.fit(X_train_scaled, Y_train)
        Y_est = regressor.predict(X_test_scaled)

        print('MAE=', mean_squared_error(Y_test, Y_est))
```

```
MAE= 0.4223090293951647
```

```
In [11]: # non linear feature transformaion StandardScaler
         # there is significant difference between price of house with 1 and with 3 to 5 inhab
         # but that diference decrese inversely proportional to numbers of inhabitants

         non_linear_feat = 5 # number of inhabitants
         X_train_new_feat = np.sqrt(X_train[:,non_linear_feat])
         X_train_new_feat.shape = (X_train_new_feat.shape[0], 1)
         X_train_extended = np.hstack([X_train, X_train_new_feat])

         X_test_new_feat = np.sqrt(X_test[:,non_linear_feat])
         X_test_new_feat.shape = (X_test_new_feat.shape[0], 1)
         X_test_extended = np.hstack([X_test, X_test_new_feat])

         scaler = StandardScaler()
         X_train_extended_scaled = scaler.fit_transform(X_train_extended)
         X_test_extended_scaled = scaler.fit_transform(X_test_extended)

         regressor = KNeighborsRegressor()
         regressor.fit(X_train_extended_scaled, Y_train)
         Y_est = regressor.predict(X_test_extended_scaled)

         print('MAE=', mean_squared_error(Y_test, Y_est))
```

```
MAE= 0.42601025900283035
```

```
In [12]: # non linear feature transformaion RobustScaler
         # there is significant difference between price of house with 1 and with 3 to 5 inhab
         # but that diference decrese inversely proportional to numbers of inhabitants

         non_linear_feat = 5 # number of inhabitants
```

```
        X_train_new_feat = np.sqrt(X_train[:,non_linear_feat])
        X_train_new_feat.shape = (X_train_new_feat.shape[0], 1)
        X_train_extended = np.hstack([X_train, X_train_new_feat])

        X_test_new_feat = np.sqrt(X_test[:,non_linear_feat])
        X_test_new_feat.shape = (X_test_new_feat.shape[0], 1)
        X_test_extended = np.hstack([X_test, X_test_new_feat])

        scaler = RobustScaler()
        X_train_extended_scaled = scaler.fit_transform(X_train_extended)
        X_test_extended_scaled = scaler.fit_transform(X_test_extended)

        regressor = KNeighborsRegressor()
        regressor.fit(X_train_extended_scaled, Y_train)
        Y_est = regressor.predict(X_test_extended_scaled)

        print('MAE=', mean_squared_error(Y_test, Y_est))
```

```
MAE= 0.42983392935256004
```

In [ ]: