Pre-processing and cleaning data: Predicting House Prices using Machine Learning

A Case Study of Saudi Arabia

1. Importing the Libraries

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load the dataset

In [ ]:
```
AQAR_Data = pd.read_csv(r"../../DataSet/aqarItemsData19971.csv")
df = AQAR_Data.copy()
```

3. Drop Features

We have some faetures that we want to drop because they do not add any value to the project like:

- id
- title
- description
- ad_number
- last_update
- views

In [ ]:
```
df.drop(columns=['id','title','description','ad_number','last_update','views'], axis=1, inplace = True)
```

Dataframe after droping features:

In [ ]:
```
df.head()
```

Out[ ]:

	city	district	front	rooms	living_rooms	bath_rooms	street_width	level	age	kitchen	garage	elevator	area	dimensions	price
0	جدة	حي الروابي	شمالية غربية	4	NaN	3	15م	4	8	1.0	1.0	1.0	103 م²	11م طول - 13م عرض7	420,000 ريال
1	الدمام	حي الفردوس	جنوبية	3	2.0	3	5م	أرضي	جديد	1.0	1.0	1.0	144 م²	NaN	610,000 ريال
2	ابها	حي القريقر	شرقية	2	1.0	2	20م	أول	جديد	1.0	NaN	1.0	340 م²	100م طول - 3م عرض	960,000 ريال
3	الدمام	حي النور	جنوبية غربية	5	1.0	3	40م	NaN	جديد	NaN	1.0	1.0	182 م²	18م طول - 10م عرض	500,000 ريال
4	الدمام	حي النور	شمالية غربية	5	1.0	3	30م	أرضي	جديد	1.0	NaN	NaN	197 م²	5م طول - 39م عرض	580,000 ريال

4. Fix missing data

We have some missing data and we want to fix this data.

All these values will be replaced to 0 or 1 by default. The 1 means that all this data is in the houses, The 0 means that all this data is not in the houses,and it was not mentioned in the AD

In [ ]:
```
df['level'].fillna(value = 0, inplace = True)
df['age'].fillna(value = 0, inplace = True)
df['kitchen'].fillna(value = 1, inplace = True)
df['garage'].fillna(value = 0, inplace = True)
df['elevator'].fillna(value = 0, inplace = True)
df['living_rooms'].fillna(value = 0, inplace = True)
df['bath_rooms'].fillna(value = 1, inplace = True)
```

4.1 fix the `area` feature

In [ ]:
```
df['area'] = df['area'].str.split().str[0]
df['area'] = df['area'].str.replace(',','')
```

```
df['area'] = df['area'].str.replace('0','²م')
df['area'] = df['area'].astype('int')
df['area'].unique()
```

Out[ ]:
```
array([ 103,  144,  340,  182,  197,  112,  297,  179,  131,  230,  252,
        472,  194,  175,  100,  166,  135,  180,  280,  105,  200,   97,
        128,  140,  161,  190,  185,  130,  164,  173,  220,  170,  139,
        215,  232,  120,  118,  210,  138,  129,   98,  177,  235,  160,
        186,  163,  300,  147,  221,  400,  142,  266,  370,  188,  145,
        218,  125,  273,  165,  150,  134,  110,  650,  315,  151,  250,
        245,  121,  146,  260,  176,  263,  171,  183,  155,  265,  189,
        225,  240,  109,  251,  159,  133,  270,  132,  202,  174,  330,
        122,  213,  238,  275,  115,  169,  700,  600,  167,  192,  196,
        127,  119,  219,  195,  191,  216,  141,  172,  325,  207,   20,
        148,  113,  227,  223,  153,  241,  310,  162,  136,  312,  158,
        258,   96,   88,  152,   81,  168,  198,  247,  209,  205,  203,
        303,  156,  262,  224,  305,  212,  107,  350,   93,  281,  217,
        450,  178,  380,  289,  199,  114,  204,  104,  106,  126,  233,
        246,   61,  154,  211,  231,  208,  272,  111,  143,  339,   94,
         35,  123,  149,  378,  193,  267,  187,  236,  137,  320,  316,
        255,   80,  117,  244,  116,  365,  214,  201,  256,  248,  157,
        326,  242,  226,  229, 1000,  264,  276,  500,  285,  295,  124,
        102,  290,  184,  108,  234,  420,  327,  253,  206,  523,  274,
        259,  288,   92,  222,  800,  550,  257,  317,  360,  243,  279,
         45,  313,  254,  283,  358,  228,   95,  385,   90,  302,   62,
         85,  640,  345,  294,  237,  377,  269,  282,   74,   82,   76,
        278,   46,  181,   59,  390,  750,  239,  406,  900,   22,  495,
        560,   54,  630,   73,   38,  399,  526,  101,  277,  291,  775,
         89,  371,  520,  375,   79,  307,  532,  261,  570,  249,  301,
        293,  332,   84,  670,   86,  308,   39,   40,  306,  428,  591,
        485,  486,   50,   75,  649,  475,  271,  299,  391,  474,   70,
        575,  268,  284,  425,  437,  430,  311,  321,  318,  529,   60,
         37,  660,  740,   87,  314,  337,  656,  352,  459,  395,   30,
        460,   41,   99,  296,    0,  572,  361,  287,   42,  440,  504,
         69,   24,  409,  335,  331,  490,  441,  436,  323,  338,  445,
        324,  343,  625,  616,   72,  344,  298,   55,   25,  292,  666,
        356,   91,  597,  408,  960,  364,   83,  353])
```

4.2 fix the `room` feature

In [ ]:
```
df[['rooms']] = df[['rooms']].fillna('0')
df.loc[df['rooms']  == '7+', 'rooms'] = '7'
df['rooms'] = df['rooms'].astype('int')
df['rooms'].unique()
```

Out[ ]:
```
array([4, 3, 2, 5, 6, 7, 1, 0])
```

4.3 fix the `front` feature

In [ ]:
```
df[['front']] = df[['front']].fillna('')
df.loc[df['front'] == 'شمالية', 'front'] = 'North'
df.loc[df['front'] == 'جنوبية', 'front'] = 'South'
df.loc[df['front'] == 'غربية', 'front'] = 'West'
df.loc[df['front'] == 'شرقية', 'front'] = 'East'
df.loc[df['front'] == 'شمالية شرقية', 'front'] = 'North-East'
df.loc[df['front'] == 'شمالية غربية', 'front'] = 'North-West'
df.loc[df['front'] == 'جنوبية شرقية', 'front'] = 'South-East'
df.loc[df['front'] == 'جنوبية غربية', 'front'] = 'South-West'
df.loc[df['front'] == 'ثلاث شوارع', 'front'] = 'Three-Streets'
df.loc[df['front'] == 'أربع شوارع', 'front'] = 'Four-Streets'
df['front'].unique()
```

Out[ ]:
```
array(['North-West', 'South', 'East', 'South-West', 'West', 'North',
       'South-East', 'North-East', 'Three-Streets', 'Four-Streets', ''],
      dtype=object)
```

4.4 fix the `living-room` feature

In [ ]:
```
df['living_rooms'] = df['living_rooms'].astype('int')
df['living_rooms'].unique()
```

Out[ ]:
```
array([0, 2, 1, 3, 5, 4, 6])
```

4.5 fix the `bath_rooms` feature

In [ ]:
```
df.loc[df['bath_rooms']  == '5+', 'bath_rooms'] = '5'
df['bath_rooms'] = df['bath_rooms'].astype('int')
df['bath_rooms'].unique()
```

Out[ ]:
```
array([3, 2, 5, 4, 1])
```

4.6 fix the `street_width` feature

In [ ]:
```
df['street_width'] = df['street_width'].str.replace('م', '')
df[['street_width']] = df[['street_width']].fillna('0')
df['street_width'] = df['street_width'].astype('int')
df['street_width'].unique()
```

Out[ ]:
```
array([ 15,   5,  20,  40,  30, 100,  24,  32,  16,  36,   0,  50,  18,
        25,  21,  23,  33,  12,  35,  26,  22,  52,  10,  19,  57,  34,
        11,  29,  17,  31,  13,   9,  28,  45,  37,  14,  27,  39,  60,
        41,  38,  86,  70,  71,   7,  62,  56,   1,  67,   6,  44,  80,
        82,   4,  87,  81,   8,  65,  58,  51,  61,  53,  55,  54,  59,
        63,  64,  89,  74,  43,  42,  96,  76,  72,  84,  73,  88,   3,
        69,  66,  90,  98])
```

4.7 fix the `level` feature

In [ ]:
```
df.loc[df['level'] == '20+', 'level'] = '20'
df.loc[df['level'] == 'أول', 'level'] = '1'
df.loc[df['level'] == 'أرضي', 'level'] = '0'
df['level'] = df['level'].astype('int')
df['level'].unique()
```

Out[ ]:
```
array([ 4,  0,  1, 16,  3,  5,  6,  7, 13, 20, 11,  8, 18, 17, 12,  9, 14,
       10, 19])
```

4.8 fix the `age` feature

In [ ]:
```
df.loc[df['age'] == 'جديد', 'age'] = '0'
df.loc[df['age'] == '35+', 'age'] = '35'
df['age'] = df['age'].astype('int')
df['age'].unique()
```

Out[ ]:
```
array([ 8,  0,  1,  9,  7, 12, 15, 11,  2, 10,  6, 14,  5,  3,  4, 25, 30,
       17, 16, 13, 33, 18, 20, 29, 35, 28, 32, 19])
```

4.9 fix the `kitchen` feature

In [ ]:
```
df['kitchen'] = df['kitchen'].astype('int')
df['kitchen'].unique()
```

Out[ ]:
```
array([1])
```

4.10 fix the `garage` feature

In [ ]:
```
df['garage'] = df['garage'].astype('int')
df['garage'].unique()
```

Out[ ]:
```
array([1, 0])
```

4.11 fix the `elevator` feature

In [ ]:
```
df['elevator'] = df['elevator'].astype('int')
df['elevator'].unique()
```

Out[ ]:
```
array([1, 0])
```

4.11 fix the `price` feature

In [ ]:
```
df['price'].fillna(value = np.nan, inplace = True)
df['price'] = df['price'].str.replace('ريال', '')
df['price'] = df['price'].str.replace('شهري', '')
df['price'] = df['price'].str.replace('سنوي', '')
df['price'] = df['price'].str.replace('/', '')
df['price'] = df['price'].str.replace(',', '')
df['price'] = df['price'].str.strip()
df = df[df['price'].notna()]
df['price'] = df['price'].astype('int')
df['price'].unique()
```

Out[ ]:
```
array([  420000,   610000,   960000,   500000,   580000,   335000,
         950000,   680000,   900000,   780000,   850000,  5150000,
         720000,   490000,   360000,   520000,  1020000,  1100000,
         690000,   745000,   499000,   600000,   990000,   751000,
         290000,   550000,   530000,   440000,   450000,   979000,
         800000,   750000,   560000,  1145000,    50000,   670000,
         620000,   350000,   909000,   319000,   700000,   540000,
         545000,   740000,  1450000,   980000,   470000,   570000,
         770000,   930000,  1000000,   485000,  8300000,   400000,
         820000,   430000,  1250000,   410000,   959000,   460000,
         640000,   825000,   940000,  1189000,   650000,   730000,
         535000,   480000,   465000,  9019000,   445000,   615000,
         590000,   675000,   475000,   899000,   710000,   619000,
         760000,   830000,  1400000,   795000,  3800000,   749000,
         951000,   459000,   890000,   660000,   320000,   790000,
        1330002,   870000,   591000,   840000,   999000,  1092000,
         920000,   399000,   970000,  1500000,   310000,   340000,
        1200000,   390000,   435000,   575000,   469000,  1800000,
         585000,   395000,   330000,   860000,   755000,   380000,
         880000,  1089000,  1220000,  1350000,   370000,   630000,
        1030000,  1247000,   659000,  1119300,   510000,   300000,
        1300000,   375000,   439000,  1139000,  1360000,  1780000,
        2000000,  1150000,   569000,  1449000,   605000,   495000,
         839000,   595000,  1270000,  1140000,   327000,  1750000,
        1120000,   365000,  1050000,   525000,   779000,  1340000,
         299000,  1375000,  1035000,   429000,   869000,   250000,
         625000,  1055000,  1060000,  1180000,  1014000,  1289000,
         985000,  1225000,   911000,  1070000,   538000,   449000,
         685000,   645000,   735000,   425000,  1015000,  1029000,
         699000,   240000,   900080,   890110,   975000,   715000,
         799000,   572000,  1900000,   513000,   385000,   210000,
         815000,   875000,   165000,   505000,   500010,  1258000,
         845000,  1039000,   797000,  1165000,   665000,  1490000,
         230000,   925000,   686000,   769000,   865000,   939000,
        1330000,  1190000,   609000,   555000,  4300000,   810000,
          65000,   279000,   200000,  1339000,   275000,  5000000,
        1249000,   664000,  2100000,   265000,   100000,   500950,
         478000,   280000,  2600000,  1280000,   802425,   720006,
        1650000,  1690000,   695000,   455000,  1170000,  4750000,
         910000,  1700000,   775000,  2300000,   725000,   887000,
         497000,   427000,   635000,  3000000,   405000,   655000,
         270000,   515000,   598000,   565000,  1430000,   415000,
        1890000,   626000,  1580000,  1169000,   849000,  1217000,
        8000000,   984000,   955000,  1850000,   918000,   599000,
         705000,   929000,   835000,   729000,   829000,   935000,
        1600000,   905000,  1299000,   897000,  1199000,  1237000,
         899999,  1369000,  1310000,  6000000,   915000,  1295000,
        1275000,  1135000,  1699000,   149000,  1080000,  1045000,
         479000,  1385000,   765000,   720154,   627000,  8500000,
         719000,   720586,   456000,   785000,  1198000,  9000000,
        1065000,  2500000,   854000,  1025222,   895000,   455555,
         220000,   409000,   720052,  1510000,   945000,  1069000,
        1530000,  1219000, 10000000,  1259000,  1090000,   844000,
         325000,  8620000,  1130000,   965000,  1477931,   666000,
         842000,   576660,  1755000,  4000000,   150000,  1495000,
        1290000,   444000,  7000000,  1550000,   533950,  1287000,
         767000,   874000,   994500,  1296000,   895800,  1222000,
         911430,  1005000,  1317000,   923000,   994250,  1012000,
         956000,   874120,   898000,  1105000,  1119000,  1175000,
         900770,   309000,   885000,   969000,  1012700,  1379000,
         889000,  1049000,  1455000,  1098112,   739000,   419000,
         549000,   269000,   989000,   898960,   446000,   199000,
         489000,  1255000,  1254000,  1599000,   919000,   853000,
         669000,   738000,  2550000,   855000,  5950000,  4500000,
         811000,   979800,   998000,   995000,   853415,   992000,
         958000,   632000,  1010000,  4600000,  8000060,  1285000,
         879000,  1230000,  1160000,   912000,  1390000,   649000,
        1000050,  1630000,  1054500,   153000,  1015500,  1013000,
        1019000,  3600000,  6500000,  2400000,   819000,  1429000,
        2800000,  6000700,   260000,  1380000,  2900000,  1099000,
        1480000,  1395000,  2388500,   768000,   809000,   289000,
         731000,   949000,   891000,    72000,   589000,   295000,
         851000,   536000,  1980000,   293000,   629000,  1110000,
        3200000,   703000,  1260000,   581000,  4800000,  2299000,
        3500000,  3550000,   355000,  1025000,   937000,  1149000,
        5020000,   656000,   757000,   582000,   379000,  1460000,
         533000,   611000,  4290000,   688000,   345000,   638000,
        1240000,   687000,   593000,   587000,   740008,   750002,
         708000,  1399000,   714000,  7500000,  4890000,   315000,
         652000,    74000,   822000,   596000,   646000,  1950000,
        4950000,   654000,   471000,   612000,   541000,   683000,
         592000,  3250000,   878000,   313553,  4850000,  3350000,
         657000,  1595000,   718521,   424091,   255000,   651000,
```

```
 639000,   780008,  1059000,  2700000,  1205000,   978500,
1236000,      590,  1114000,   602000,   329000,  1372000,
1129000,   843000,  1316000,  1349999,   433000,   474000,
 689000,   900050,  1173000,  1144000,   369000,   852000,
 170000, 13000000,  2200000,   509000,  1239000,  1000500,
    700,      430,   494000,  1075000,      460,   486000,
1075269,  9070000,  1115000,   814000,   848000,   285000,
    440,      620,  1489000,      450,   523000,   504400,
8490000,  1499000,  1549000,   859000,  1204000,  1619000,
2850000,   451000,  4200000,   399999,   566000])
```

4.12 fix the `dimensions` feature

In [ ]:
```
df['dimensions'] = df['dimensions'].str.findall(r"\d+")
df['dimensions'] = df[['dimensions']].fillna('0')
df['dimensions']
```

Out[ ]:
```
0         [7, 13]
1               0
2        [3, 100]
3        [10, 18]
4         [39, 5]
           ...
19966           0
19967    [17, 10]
19968           0
19969     [2, 86]
19970     [21, 8]
Name: dimensions, Length: 19956, dtype: object
```

Dataframe after fixing missing data:

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
city            0
district        0
front           0
rooms           0
living_rooms    0
bath_rooms      0
street_width    0
level           0
age             0
kitchen         0
garage          0
elevator        0
area            0
dimensions      0
price           0
dtype: int64
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19956 entries, 0 to 19970
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   city          19956 non-null  object
 1   district      19956 non-null  object
 2   front         19956 non-null  object
 3   rooms         19956 non-null  int32
 4   living_rooms  19956 non-null  int32
 5   bath_rooms    19956 non-null  int32
 6   street_width  19956 non-null  int32
 7   level         19956 non-null  int32
 8   age           19956 non-null  int32
 9   kitchen       19956 non-null  int32
 10  garage        19956 non-null  int32
 11  elevator      19956 non-null  int32
 12  area          19956 non-null  int32
 13  dimensions    19956 non-null  object
 14  price         19956 non-null  int32
dtypes: int32(11), object(4)
memory usage: 1.6+ MB
```

5. Removing Outliers

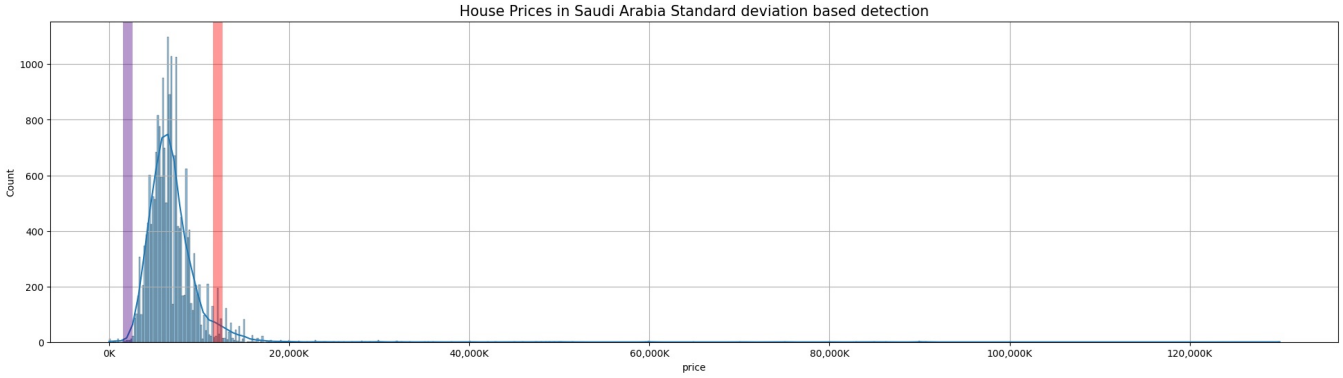
Outliers are data points that exist far away from the majority of your data. This can happen due to several reasons, such as incorrect data recording to genuine rare occurrences. Either way you will often want to remove these values as they can negatively impact your models.

An example of the negative effect can be seen here where an outlier is causing almost all of the scaled data to be squashed to the lower bound.

5.1 Target Variable "price" Visualization with Outliers boundaries using Standrad Deviation

In [ ]:
```
# Making a histogram of the price variable and adding Standard deviation based detection"
f, ax = plt.subplots(figsize = (24,6))
ax = sns.histplot(x= df['price'], kde= True)

# choose 1.7 standard deviation from the mean:
factor = 1.2

# define upper:
upper_lim = df['price'].mean () + df['price'].std () * factor

# define lower:
lower_lim = df['price'].mean () - df['price'].std () * factor

#Plot upper and lower based on 1.2 standard deviation from the mean:
ax.axvline(upper_lim, color='red', ls='-', alpha=0.4, lw=10)
ax.axvline(lower_lim, color='indigo', ls='-', alpha=0.4, lw=10)
ax.set_title('House Prices in Saudi Arabia Standard deviation based detection', fontsize=15)
xlabels = ['{:,.0f}'.format(x) + 'K' for x in ax.get_xticks()/100]
ax.set_xticklabels(xlabels)
plt.grid(True)
```

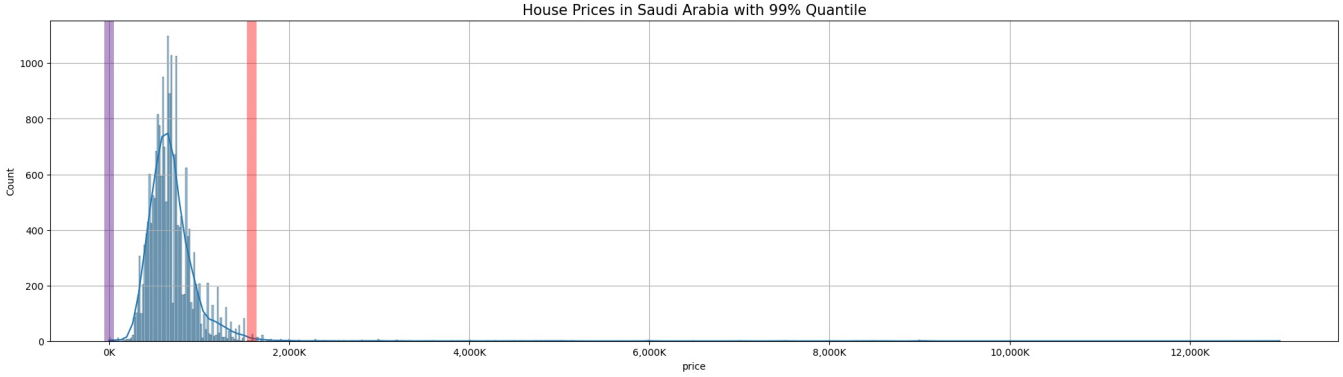
```
C:\Users\hasoo\AppData\Local\Temp\ipykernel_47252\479889413.py:19: UserWarning: FixedFormatter should only be u
sed together with FixedLocator
  ax.set_xticklabels(xlabels)
```



5.2 Target Variable "price" Visualization with Outliers boundaries using Quantile

In [ ]:
```
# Making a histogram of the price variable and Quantile based detection
f, ax = plt.subplots(figsize = (24,6))
ax = sns.histplot(x=df['price'], kde=True)
# define upper using 99th quantile:
upper_lim = df['price'].quantile(0.99)
# upper quantile:
lower_lim = df['price'].quantile(0)
#Plot upper and lower based on 0th, 99th quantile:
ax.axvline(upper_lim, color='red', ls='-', alpha=0.4, lw=10)
ax.axvline(lower_lim, color='indigo', ls='-', alpha=0.4, lw=10)
ax.set_title('House Prices in Saudi Arabia with 99% Quantile', fontsize=15)
xlabels = ['{:,.0f}'.format(x) + 'K' for x in ax.get_xticks()/1000]
ax.set_xticklabels(xlabels)
plt.grid(True)
```

```
C:\Users\hasoo\AppData\Local\Temp\ipykernel_47252\1611047075.py:13: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels(xlabels)
```



5.3 Outliers: Standard deviation based detection

5.3 Outliers: Standard deviation based detection

In [ ]:
```
factor = 1.2
upper_lim = df['price'].mean () + df['price'].std () * factor
lower_lim = df['price'].mean () - df['price'].std () * factor

df_std = df[(df['price'] < upper_lim) & (df['price'] > lower_lim)]
```

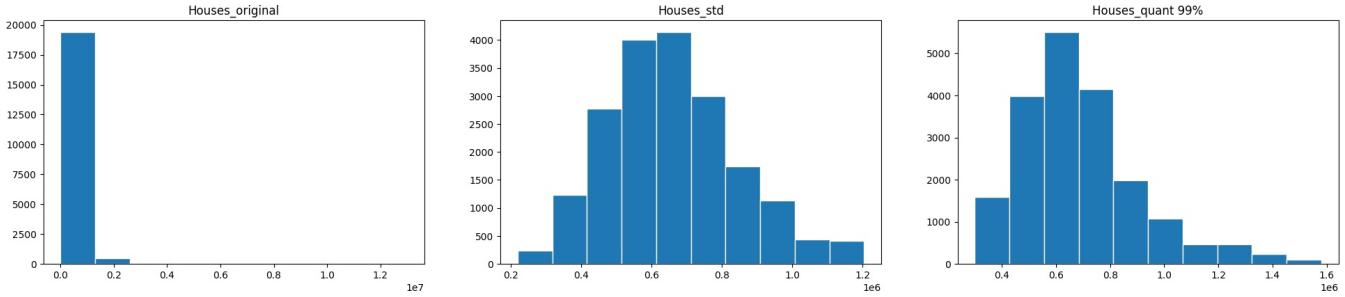
5.4 Outliers: Quantile based detection

In [ ]:
```
upper_lim = df['price'].quantile(0.99)
lower_lim = df['price'].quantile(0.01)

df_quant = df[(df['price'] < upper_lim) & (df['price'] > lower_lim)]
```

5.5 Visualization all outliers: Target Variable "price" Visualization

In [ ]:
```
plt.subplots(figsize = (24,10))
plt.subplot(2,3,1)
plt.hist(df[['price']], edgecolor='white')
plt.title('Houses_original')
plt.subplot(2,3,2)
plt.hist(df_std[['price']], edgecolor='white')
plt.title('Houses_std')
plt.subplot(2,3,3)
plt.hist(df_quant[['price']], edgecolor='white')
plt.title('Houses_quant 99%')
plt.show()
```

```
C:\Users\hasoo\AppData\Local\Temp\ipykernel_47252\4035440049.py:2: MatplotlibDeprecationWarning: Auto-removal o
f overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.rem
ove() as needed.
  plt.subplot(2,3,1)
```



In [ ]:
```
sns.boxplot(df['price'])
```

Out[ ]:
```
<AxesSubplot: >
```