In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from catboost import CatBoostRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from lightgbm import LGBMRegressor

import warnings
warnings.filterwarnings('ignore')
```

```
/usr/lib/python3/dist-packages/pkg_resources/__init__.py:116: PkgResource
sDeprecationWarning: 1.16.0-unknown is an invalid version and will not be
supported in a future release
  warnings.warn(
/usr/lib/python3/dist-packages/pkg_resources/__init__.py:116: PkgResource
sDeprecationWarning: 1.1build1 is an invalid version and will not be supp
orted in a future release
  warnings.warn(
/usr/lib/python3/dist-packages/pkg_resources/__init__.py:116: PkgResource
sDeprecationWarning: 0.1.43ubuntu1 is an invalid version and will not be
supported in a future release
  warnings.warn(
```

In [5]:
```
train = pd.read_csv("Housing_dataset_train.csv")
test = pd.read_csv("Housing_dataset_test.csv")
sub = pd.read_csv("Sample_submission.csv")
#var = pd.read_csv("VariableDefinitions.csv")
```

In [6]:
```
train.head()
```

Out[6]:

	ID	loc	title	bedroom	bathroom	parking_space	price
0	3583	Katsina	Semi-detached duplex	2.0	2.0	1.0	1149999.565
1	2748	Ondo	Apartment	NaN	2.0	4.0	1672416.689
2	9261	Ekiti	NaN	7.0	5.0	NaN	3364799.814
3	2224	Anambra	Detached duplex	5.0	2.0	4.0	2410306.756
4	10300	Kogi	Terrace duplex	NaN	5.0	6.0	2600700.898

In [9]:
```
train[train['price'] > 0].reset_index(drop = True)
```

Out[9]:

	ID	loc	title	bedroom	bathroom	parking_space	price
0	3583	Katsina	Semi-detached duplex	2.0	2.0	1.0	1149999.565
1	2748	Ondo	Apartment	NaN	2.0	4.0	1672416.689
2	9261	Ekiti	NaN	7.0	5.0	NaN	3364799.814
3	2224	Anambra	Detached duplex	5.0	2.0	4.0	2410306.756
4	10300	Kogi	Terrace duplex	NaN	5.0	6.0	2600700.898
...	...	...	...	...	...	...	...
13995	6175	Edo	Bungalow	NaN	7.0	NaN	2367927.861
13996	9704	Kaduna	Apartment	NaN	7.0	5.0	2228516.471
13997	11190	Plateau	Bungalow	8.0	6.0	5.0	2406812.693
13998	9256	Delta	Flat	NaN	6.0	1.0	3348918.718
13999	8787	Nasarawa	NaN	9.0	7.0	5.0	2858516.890

14000 rows × 7 columns

In [11]:
```
train.describe()
```

Out[11]:

	ID	bedroom	bathroom	parking_space	price
count	14000.000000	12201.000000	12195.000000	12189.000000	1.400000e+04
mean	4862.700357	4.308171	3.134235	3.169825	2.138082e+06
std	3818.348214	2.441165	2.035950	1.599415	1.083057e+06
min	0.000000	1.000000	1.000000	1.000000	4.319673e+05
25%	1672.750000	2.000000	1.000000	2.000000	1.393990e+06
50%	3527.000000	4.000000	2.000000	3.000000	1.895223e+06
75%	8011.250000	6.000000	5.000000	4.000000	2.586699e+06
max	12999.000000	9.000000	7.000000	6.000000	1.656849e+07

In [12]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14000 entries, 0 to 13999
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   ID             14000 non-null  int64  
 1   loc            12187 non-null  object 
 2   title          12278 non-null  object 
 3   bedroom        12201 non-null  float64
 4   bathroom       12195 non-null  float64
 5   parking_space  12189 non-null  float64
 6   price          14000 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 765.8+ KB
```

In [6]:
```
test.head()
```

Out[6]:

	ID	loc	title	bedroom	bathroom	parking_space
0	845	Kano	Penthouse	4	1	2
1	1924	Adamawa	Apartment	2	2	4
2	10718	Adamawa	Bungalow	2	7	2
3	12076	Lagos	Mansion	9	5	2
4	12254	Gombe	Semi-detached duplex	5	6	1

In [13]:
```
sub.head()
```

Out[13]:

	ID
0	845
1	1924
2	10718
3	12076
4	12254

In [17]:
```
train.title.unique()
```

Out[17]:
```
array(['Semi-detached duplex', 'Apartment', nan, 'Detached duplex',
       'Terrace duplex', 'Mansion', 'Bungalow', 'Penthouse', 'Townhouse',
       'Flat', 'Cottage'], dtype=object)
```

In [18]:
```
data = train.drop("price", axis=1).append(test)
```

In [19]:

```
# Define the ranking based on size (arranged from smallest to biggest)
house_type_ranks = {
    'Cottage': 1,
    'Bungalow': 2,
    'Townhouse': 3,
    'Terrace duplex': 4,
    'Detached duplex': 5,
    'Semi-detached duplex': 6,
    'Flat': 7,
    'Penthouse': 8,
    'Apartment': 9,
    'Mansion': 10
}

# Map the house types to numerical values based on size ranking
data['title'] = data['title'].map(house_type_ranks)

# Print the updated dataframe
data.head()
```

Out[19]:

	ID	loc	title	bedroom	bathroom	parking_space
0	3583	Katsina	6.0	2.0	2.0	1.0
1	2748	Ondo	9.0	NaN	2.0	4.0
2	9261	Ekiti	NaN	7.0	5.0	NaN
3	2224	Anambra	5.0	5.0	2.0	4.0
4	10300	Kogi	4.0	NaN	5.0	6.0

In [21]:

```
# Calculate the frequency of each category in the 'loc' column
category_frequencies = data['loc'].value_counts(normalize=True)

# Create a dictionary to map each category to its corresponding frequency
loc_frequency_mapping = category_frequencies.to_dict()

# Map the 'loc' column to its corresponding frequency values
data['loc'] = data['loc'].map(loc_frequency_mapping)

# Print the updated dataframe
data.head()
```

Out[21]:

	ID	loc	title	bedroom	bathroom	parking_space
0	3583	0.028592	6.0	2.0	2.0	1.0
1	2748	0.028097	9.0	NaN	2.0	4.0
2	9261	0.027822	NaN	7.0	5.0	NaN
3	2224	0.029856	5.0	5.0	2.0	4.0
4	10300	0.027382	4.0	NaN	5.0	6.0

In [22]:
```
train.shape[0]
```

Out[22]: 14000

In [23]:
```
X = data[:train.shape[0]]
y = train.price
test_df = data[train.shape[0]:]
```

In [24]:
```
X.shape, y.shape, test_df.shape
```

Out[24]: ((14000, 6), (14000,), (6000, 6))

In [25]:
```
sns.set()
y_transformed = pd.Series(np.log1p(y))

fig, ax = plt.subplots(1, 2)
sns.distplot(y, ax=ax[0])
plt.title("Target after Log transformation")
# ax[0].axvline(y_transformed)
sns.distplot(y_transformed, ax=ax[1])
plt.show()
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
```

In [28]:
```
lgb = LGBMRegressor()
lgb.fit(X_train, y_train)
lgb_preds = lgb.predict(X_test)

print(f'mse = {mean_squared_error(y_test, lgb_preds, squared=False)}')
```

```
mse = 592295.8375673272
```

In [29]:
```
predictions = lgb.predict(test_df)
predictions
```

Out[29]:
```
array([2315313.77836646,  847985.77448807, 1116418.07946145, ...,
       2018114.35331414, 1420440.4714651 , 3168784.09197576])
```

In [31]:
```
sub['price'] = predictions
sub.head()
```

Out[31]:

	ID	price
0	845	2.315314e+06
1	1924	8.479858e+05
2	10718	1.116418e+06
3	12076	7.349789e+06
4	12254	1.966154e+06

In [32]:
```
sub.to_csv('baseline model.csv', index=False)
```

In [ ]:

In [ ]:

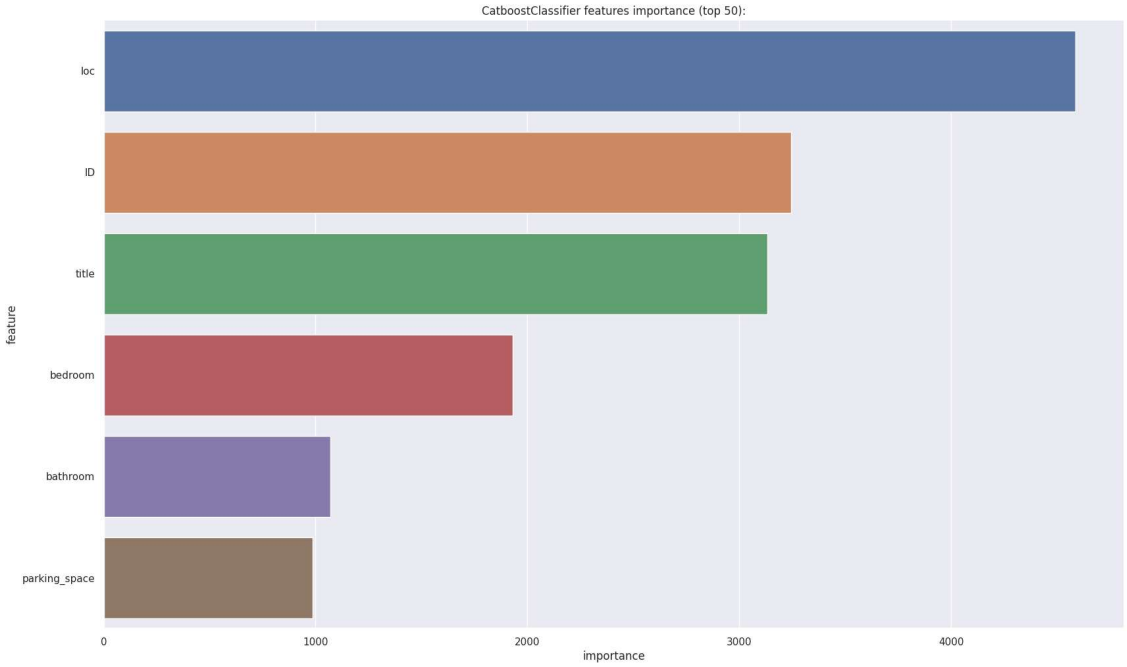
In [33]:

```
fold_pred=[]
oof_pred = []

params = {
    'n_estimators': 500,
    'colsample_bytree': 0.86,
 'learning_rate': 0.032,
 'max_depth': 7,
 'subsample': 0.85}


fold = KFold(n_splits=7, shuffle=True)#15#5#10
i=1
for train_index, test_index in fold.split(X,y):

    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = np.log1p(y.iloc[train_index]), y.iloc[test_index]

    model = LGBMRegressor(**params, objective = "rmse")
    model.fit(X_train,y_train,eval_set=[(X_train,y_train),(X_test, y_test)

    preds= model.predict(X_test)
    print("err: ",(mean_squared_error(y_test,np.expm1(preds), squared=Fals
    oof_pred.append(mean_squared_error(y_test,np.expm1(preds),squared=Fals
    p2 = model.predict(test_df[X.columns])
    fold_pred.append(np.expm1(p2))


print(np.mean(oof_pred))
```

```
err:  584931.2949560428
err:  454369.0911503518
err:  630908.0483883314
err:  514765.70916750474
err:  516133.18948320346
err:  586472.4562380317
err:  569293.8552481126
550981.9492330827
```

In [34]:
```
# # 0.7831404345536787
feature_importance_df = pd.DataFrame(model.feature_importances_, columns=[
feature_importance_df['feature'] = X.columns

plt.figure(figsize=(20, 12));
sns.barplot(x="importance", y="feature", data=feature_importance_df.sort_v
plt.title('CatboostClassifier features importance (top 50):')
```

Out[34]: Text(0.5, 1.0, 'CatboostClassifier features importance (top 50):')



In [35]:
```
sub.head()
```

Out[35]:

	ID	price
0	845	2.315314e+06
1	1924	8.479858e+05
2	10718	1.116418e+06
3	12076	7.349789e+06
4	12254	1.966154e+06

In [37]:
```
sub['price'] =  np.mean(fold_pred, axis = 0)
```

In [38]:
```
sub.head()
```

Out[38]:

	ID	price
0	845	2.306484e+06
1	1924	9.169233e+05
2	10718	1.128666e+06
3	12076	7.984726e+06
4	12254	1.948246e+06

In [39]:
```
sub.to_csv('LGB kfold_7 Baseline submission.csv', index=False)
```

In [ ]:
```

```