In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import make_scorer
import random
import collections
%matplotlib inline
```

```
/opt/conda/lib/python3.9/site-packages/scipy/__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (det
ected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Adatelőkészítés

In [2]:

```
df = pd.read_csv('data/DataSet_LakasArak.csv')
```

In [3]:

```
del df['ad_view_cnt']
del df['active_days']
del df['nr']
```

In [4]:

```
random.seed(8594726138)
columns = list(df.columns)
random.shuffle(columns)
df = df[columns]
df
```

Out[4]:

	elevator_type	county	room_cnt	view_type	balcony_area	garden_access	created_
0	yes	Budapest	2.0	street view	0.0	NaN	2015-0
1	yes	Budapest	1.0	street view	0.0	NaN	2015-0
2	yes	Budapest	2.0	garden view	0.0	NaN	2015-0
3	none	Budapest	2.0	garden view	4.0	NaN	2015-0
4	yes	Budapest	2.0	NaN	3.0	NaN	2015-0
...	...	...	...	...	...	...	
78534	none	Budapest	2.0	NaN	0.0	NaN	2016-0
78535	yes	Budapest	1.0	NaN	0.0	NaN	2016-0

	elevator_type	county	room_cnt	view_type	balcony_area	garden_access	created_
78536	none	Budapest	1.0	NaN	0.0	NaN	2016-0 2
78537	none	Budapest	1.0	NaN	0.0	NaN	2016-0 2
78538	yes	Budapest	2.0	street view	0.0	NaN	2016-0 2

78539 rows × 19 columns

In [5]:

```
df.columns
```

Out[5]:

```
Index(['elevator_type', 'county', 'room_cnt', 'view_type', 'balcony_area',
       'garden_access', 'created_at', 'small_room_cnt', 'orientation',
       'property_type', 'postcode', 'property_area', 'building_floor_count',
       'city', 'property_condition_type', 'property_subtype', 'heating_type',
       'property_floor', 'price_created_at'],
      dtype='object')
```

In [6]:

```
df.isna().sum()
```

Out[6]:

```
elevator_type              14151
county                         0
room_cnt                       0
view_type                  35661
balcony_area                   0
garden_access              61339
created_at                     0
small_room_cnt                 0
orientation                30892
property_type                  0
postcode                   28954
property_area                  0
building_floor_count       42110
city                         559
property_condition_type        0
property_subtype            1659
heating_type               11306
property_floor              3793
price_created_at               0
dtype: int64
```

Ennek a módszernek célja az idő dimenzió, és ezzel esetleges trend eliminálása a négyzetméterárból. Órán volt szó arról, hogy az adathalmaz születésének időszakában az ingatlanpiac erősödött, így aki irreálisan magas árat választott, azt a piac előbb-utóbb utolérte. Ez a jelenség rontja a modellünk általánosítóképességét, ezért a hagyományos négyzetméterár helyett egy jelenértékre számított négyzetméterárat számítok ki az órán tanult módon. Ez az érték minden ingatlanra az adathalmaz utolsó napjához arányosít, így a trend nem számít bele a tanulásba és a predikcióba sem, mivel ez a jelen_nmAr lesz a célváltozó.

In [7]:

```
# jelenértékű nmÁr
df['nmAr'] = df['price_created_at']*1000000/df['property_area']
df['datum'] = pd.to_datetime(df['created_at'])
stat = df.groupby('datum', as_index=False).agg({"nmAr":"median"})
```

```
df['daynum'] = (df['datum'] - df['datum'].min()).dt.days
stat['daynum'] = (stat['datum'] - stat['datum'].min()).dt.days
minlinreg = LinearRegression()
minlinreg.fit(stat[ ['daynum'] ], stat['nmAr'])
stat['trend_nmAr']= minlinreg.predict(stat[['daynum']])
del stat['nmAr']
del stat['daynum']
df = df.merge(stat, on='datum', how='left')
df['jelen_nmAr']= df['nmAr']/df['trend_nmAr']* stat['trend_nmAr'].max()
df
```

Out[7]:

	elevator_type	county	room_cnt	view_type	balcony_area	garden_access	created_
0	yes	Budapest	2.0	street view	0.0	NaN	2015-0 (
1	yes	Budapest	1.0	street view	0.0	NaN	2015-0 (
2	yes	Budapest	2.0	garden view	0.0	NaN	2015-0 (
3	none	Budapest	2.0	garden view	4.0	NaN	2015-0 (
4	yes	Budapest	2.0	NaN	3.0	NaN	2015-0 (
...	...	...	...	...	...	...	
78534	none	Budapest	2.0	NaN	0.0	NaN	2016-0 2
78535	yes	Budapest	1.0	NaN	0.0	NaN	2016-0 2
78536	none	Budapest	1.0	NaN	0.0	NaN	2016-0 2
78537	none	Budapest	1.0	NaN	0.0	NaN	2016-0 2
78538	yes	Budapest	2.0	street view	0.0	NaN	2016-0 2

78539 rows × 24 columns

Transzformációs függvények oszlopsorrendben

In [8]:

```
def tr_elevator_type(df, input_cols):
    df['elevator_type']=df['elevator_type'].apply(lambda x: 1 if str(x)[0]=='
    input_cols += ['elevator_type']
    return df, input_cols

def tr_county(df, input_cols):
    #nem rakom bele mert minden érték azonos
    return df, input_cols

def tr_room_cnt(df, input_cols):
    input_cols += ['room_cnt']
    return df, input_cols
```

```
def tr_view_type(df, input_cols):
    # dummy
    view_dummies = pd.get_dummies(df['view_type'], prefix = 'view_type=', dro
    df = df.merge(view_dummies, left_index = True, right_index = True, how =
    input_cols += list(view_dummies.columns)
    return df, input_cols

def tr_balcony_area(df, input_cols):
    input_cols += ['balcony_area']
    return df, input_cols

def tr_garden_access(df, input_cols):
    # dummy
    garden_access_dummies = pd.get_dummies(df['garden_access'], prefix = 'gar
    df = df.merge(garden_access_dummies, left_index = True, right_index = Tru
    input_cols += list(garden_access_dummies.columns)
    return df, input_cols

def tr_created_at(df, input_cols):
    df['created_at'] = pd.to_datetime(df['created_at'])
    cr_min = df['created_at'].min()
    df['created_at'] = (df['created_at'] - cr_min).dt.days
    input_cols += ['created_at']
    return df, input_cols

def tr_small_room_cnt(df, input_cols):
    input_cols += ['small_room_cnt']
    return df, input_cols

def tr_orientation_1(df, input_cols):
    # dummy
    orientation_dummies = pd.get_dummies(df['orientation'], prefix = 'orienta
    df = df.merge(orientation_dummies, left_index = True, right_index = True,
    input_cols += list(orientation_dummies.columns)
    return df, input_cols

def tr_orientation_2(df, input_cols):
    # adott?
    df['orientation_given'] = df['orientation'].apply(lambda x: 0 if x==NaN e
    input_cols += ['orientation_given']
    return df, input_cols

def tr_property_type(df, input_cols):
    # nem rakom bele mert minden érték azonos
    return df, input_cols

def tr_postcode_1(df, input_cols):
    df['postcode'] = df['postcode'].fillna(1000)
    input_cols += ['postcode']
    return df, input_cols

def tr_postcode_2(df, input_cols):
    df['postcode'] = df['postcode'].fillna(1000)
    df['postcode'].apply(lambda x: x % 1000)
    input_cols += ['postcode']
    return df, input_cols

def tr_property_area(df, input_cols):
    input_cols += ['property_area']
    return df, input_cols

def tr_building_floor_count(df, input_cols):
    df['building_floor_count'] = df['building_floor_count'].fillna('other')
    df['building_floor_count'] = df['building_floor_count'].apply(lambda x: 1
    input_cols += ['building_floor_count']
    return df, input_cols

def tr_city_1(df, input_cols):
    # kerület dummy
    city_dummies = pd.get_dummies(df['city'], prefix = 'city=', drop_first =
    df = df.merge(city_dummies, left_index = True, right_index = True, how =
    input_cols += list(city_dummies.columns)
    return df, input_cols

def tr_city_2(df, input_cols):
    # Buda (0), Pest(1) vagy Csepel (2)?
    mapping = {
        0:2, 1:0, 2:0, 3:0, 4:1, 5:1, 6:1, 7:1, 8:1, 9:1, 10:1,
        11:0, 12:0, 13:1, 14:1, 15:1, 16:1, 17:1, 18:1, 19:1, 20:1,
        21:2, 22:0, 23:1
    }
    df['ker_code']=df['postcode'].fillna(1000).apply(lambda x: int(str(x)[1:3
    stat = df.groupby('city',as_index=False).agg({'ker_code':'max'})
    stat.iloc[9,1]=10
    stat.iloc[12,1]=13
    stat.columns=['city','kerulet_sorszam']
    df=df.merge(stat,on='city',how='left')
    df['kerulet_sorszam']=df['kerulet_sorszam'].fillna(0)
    df['kerulet_csoport']=df['kerulet_sorszam'].apply(lambda x: mapping[x])
    input_cols += ['kerulet_csoport']
    return df, input_cols

def tr_city_3(df, input_cols):
    # egyszerű számérték irányítószámból és kerületnevekből
    df['ker_code']=df['postcode'].fillna(1000).apply(lambda x: int(str(x)[1:3
    stat = df.groupby('city',as_index=False).agg({'ker_code':'max'})
    stat.iloc[9,1]=10
    stat.iloc[12,1]=13
    stat.columns=['city','kerulet_sorszam']
    df=df.merge(stat,on='city',how='left')
    df['kerulet_sorszam']=df['kerulet_sorszam'].fillna(0)
    input_cols += ['kerulet_sorszam']
    return df, input_cols

def tr_property_condition_type1(df, input_cols):
    # dummy
    property_condition_type_dummies = pd.get_dummies(df['property_condition_t
    df = df.merge(property_condition_type_dummies, left_index = True, right_i
    input_cols += list(property_condition_type_dummies.columns)
    return df, input_cols

def tr_property_condition_type2(df, input_cols):
    mapping = {
        'new_construction': 5,
        'good': 4,
        'novel': 3,
        'renewed': 2,
        'medium': 1,
        'missing_info': 0,
        'under_construction': -1,
        'can_move_in': -2,
        'to_be_renovated': -3
    }
    df['property_condition_type'] = df['property_condition_type'].apply(lambd
    input_cols += ['property_condition_type']
    return df, input_cols

def tr_property_subtype(df, input_cols):
    # dummy
    property_subtype_dummies = pd.get_dummies(df['property_subtype'], prefix
    df = df.merge(property_subtype_dummies, left_index = True, right_index =
    input_cols += list(property_subtype_dummies.columns)
    return df, input_cols

def tr_heating_type_1(df, input_cols):
    # dummy
    df['heating_type']=df['heating_type'].fillna('other')
    heating_dummies = pd.get_dummies(df['heating_type'], prefix = 'heating_ty
    df = df.merge(heating_dummies, left_index = True, right_index = True, how
    input_cols += list(heating_dummies.columns)
    return df, input_cols

def tr_heating_type_2(df, input_cols):
    # mapping gyakoriság alapján
    mapping = {
        'gas furnace':10,
        'fan-coil':9,
        'tile stove (gas)':8,
        'electric':7,
        'central heating':6,
        'central heating with own meter':5,
        'other':4,
        'district heating':3,
        'gas furnace, circulating hot water':2,
        'konvection gas burner':1
    }
    df['heating_type']=df['heating_type'].fillna('other')
    df['heating_num']=df['heating_type'].apply(lambda x: mapping[x])
    input_cols += ['heating_num']
    return df, input_cols

def prop_floor(x):
    if x == 'ground floor':
        return 0
    if x == 'mezzanine floor':
        return 0.5
    if x == '10 plus':
        return 11
    if x == 'basement':
        return -1
    return x

def tr_property_floor_1(df, input_cols):
    df['property_floor'] = df['property_floor'].fillna(0).apply(lambda x: pro
    input_cols += ['property_floor']
    return df, input_cols


def tr_property_floor_2(df, input_cols):
    # dummy
    property_floor_dummies = pd.get_dummies(df['property_floor'], prefix = 'p
    df = df.merge(property_floor_dummies, left_index = True, right_index = Tr
    input_cols += list(property_floor_dummies.columns)
    return df, input_cols

def tr_price_created_at(df, input_cols):
    # ez felesleges, ebből lett a származtatott jelen_nmAr célváltozó

    return df, input_cols

def tr_which_day_1(df, input_cols):
    df['created_at'] = pd.to_datetime(df['created_at'])
    df['day_of_week'] = df['created_at'].apply(lambda x: x.weekday())
    input_cols += ['day_of_week']
    return df, input_cols

def tr_which_day_2(df, input_cols):
    df['created_at'] = pd.to_datetime(df['created_at'])
    df['day_of_week'] = df['created_at'].apply(lambda x: x.weekday())
    df['day_of_week'] = df['day_of_week'].apply(lambda x: 1 if x == 5 or x ==
    input_cols += ['day_of_week']
    return df, input_cols
```

In [9]:

```
def mean_absolute_percentage_error(y_true,y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred)/y_true))

mape_scorer = make_scorer(mean_absolute_percentage_error, greater_is_better =
```

In [10]:

```
score_lin = []
score_gbm = []
score_neu = []

def learn_step(step_index, df, bemenok, kimeno):
    linreg = LinearRegression()
    gbm = GradientBoostingRegressor(random_state=42, max_depth=5, n_estimator
    neural = MLPRegressor(learning_rate='adaptive', random_state=42, hidden_l

    g_lin = GridSearchCV(estimator=linreg, param_grid={}, cv=3, scoring=mape_
    g_lin.fit(df[bemenok], df[kimeno])
    score_lin.append( [step_index, len(bemenok), g_lin.cv_results_['mean_test

    g_gbm = GridSearchCV(estimator=gbm, param_grid={}, cv=3, scoring=mape_sco
    g_gbm.fit(df[bemenok], df[kimeno])
    score_gbm.append( [step_index, len(bemenok), g_gbm.cv_results_['mean_test

    g_neu = GridSearchCV(estimator=neural, param_grid={}, cv=3, scoring=mape_
    g_neu.fit(df[bemenok], df[kimeno])
    score_neu.append( [step_index, len(bemenok), g_neu.cv_results_['mean_test

    return g_lin.cv_results_['mean_test_score'][0], g_gbm.cv_results_['mean_t
```

Oszlopok egyenkénti hozzáadása és tanítások:

In [11]:

```
bemenok = []
kimeno = 'jelen_nmAr'


# 1 elevator_type
print('# 1 elevator_type START')
df, bemenok = tr_elevator_type(df, bemenok)
print(bemenok)
learn_step(1, df, bemenok, kimeno)
print('# 1 elevator_type END')

# 2 county
print('# 2 county START')
df, bemenok = tr_county(df, bemenok)
learn_step(2, df, bemenok, kimeno)
print('# 2 county END')
```

```
# 3 room_cnt
print('# 3 room_cnt START')
df, bemenok = tr_room_cnt(df, bemenok)
learn_step(3, df, bemenok, kimeno)
print('# 3 room_cnt END')

# 4 view_type
print('# 4 view_type START')
df, bemenok = tr_view_type(df, bemenok)
learn_step(4, df, bemenok, kimeno)
print('# 4 view_type END')

# 5 balcony_area
print('# 5 balcony_area START')
df, bemenok = tr_balcony_area(df, bemenok)
learn_step(5, df, bemenok, kimeno)
print('# 5 balcony_area END')

# 6 garden_access
print('# 6 garden_access START')
df, bemenok = tr_garden_access(df, bemenok)
learn_step(6, df, bemenok, kimeno)
print('# 6 garden_access END')


df_created_at_1 = df.copy()
df_created_at_2 = df.copy()
bem1 = bemenok.copy()
bem2 = bemenok.copy()
# 7 created_at 1
print('# 7 created_at => which_day 1 START')
df_created_at_1, bem1 = tr_which_day_1(df_created_at_1, bem1)
_,__, res1 = learn_step(7, df_created_at_1, bem1, kimeno)
print('# 7 created_at => which_day 1 END')

# 7 created_at 2
print('# 7 created_at => which_day 2 START')
df_created_at_2, bem2 = tr_which_day_2(df_created_at_2, bem2)
_,__, res2 = learn_step(7, df_created_at_2, bem2, kimeno)
print('# 7 created_at => which_day 2 END')
if (res1 > res2):
    df = df_created_at_1
    bemenok = bem1
else:
    df = df_created_at_2
    bemenok = bem2


# 8 small_room_cnt
print('# 8 small_room_cnt START')
df, bemenok = tr_small_room_cnt(df, bemenok)
learn_step(8, df, bemenok, kimeno)
print('# 8 small_room_cnt END')


df_orientation_1 = df.copy()
df_orientation_2 = df.copy()
bem1 = bemenok.copy()
bem2 = bemenok.copy()
# 9 orientation 1
print('# 9 orientation 1 START')
df_orientation_1, bem1 = tr_which_day_1(df_orientation_1, bem1)
_,__, res1 = learn_step(9, df_orientation_1, bem1, kimeno)
print('# 9 orientation 1 END')
```

```
# 9 orientation 2
print('# 9 orientation 2 START')
df_orientation_2, bem2 = tr_which_day_2(df_orientation_2, bem2)
_,__, res2 = learn_step(9, df_orientation_2, bem2, kimeno)
print('# 9 orientation 2 END')
if (res1 > res2):
    df = df_orientation_1
    bemenok = bem1
else:
    df = df_orientation_2
    bemenok = bem2


# 10 property_type
print('# 10 property_type START')
df, bemenok = tr_property_type(df, bemenok)
learn_step(10, df, bemenok, kimeno)
print('# 10 property_type END')


df_postcode_1 = df.copy()
df_postcode_2 = df.copy()
bem1 = bemenok.copy()
bem2 = bemenok.copy()
# 11 postcode 1
print('# 11 postcode 1 START')
df_postcode_1, bem1 = tr_postcode_1(df_postcode_1, bem1)
_,__, res1 = learn_step(11, df_postcode_1, bem1, kimeno)
print('# 11 postcode 1 END')

# 11 postcode 2
print('# 11 postcode 2 START')
df_postcode_2, bem2 = tr_postcode_2(df_postcode_2, bem2)
_,__, res2 = learn_step(11, df_postcode_2, bem2, kimeno)
print('# 11 postcode 2 END')
if (res1 > res2):
    df = df_postcode_1
    bemenok = bem1
else:
    df = df_postcode_2
    bemenok = bem2

# 12 property_area
print('# 12 property_area START')
df, bemenok = tr_property_area(df, bemenok)
learn_step(12, df, bemenok, kimeno)
print('# 12 property_area END')

# 13 building_floor_count
print('# 13 building_floor_count START')
df, bemenok = tr_building_floor_count(df, bemenok)
learn_step(13, df, bemenok, kimeno)
print('# 13 building_floor_count END')


df_city_1 = df.copy()
df_city_2 = df.copy()
df_city_3 = df.copy()
bem1 = bemenok.copy()
bem2 = bemenok.copy()
bem3 = bemenok.copy()
# 14 city 1
print('# 14 city 1 START')
```

```
df_city_1, bem1 = tr_city_1(df_city_1, bem1)
_,__, res1 = learn_step(14, df_city_1, bem1, kimeno)
print('# 14 city 1 END')

# 14 city 2
print('# 14 city 2 START')
df_city_2, bem2 = tr_city_2(df_city_2, bem2)
_,__, res2 = learn_step(14, df_city_2, bem2, kimeno)
print('# 14 city 2 END')

# 14 city 3
print('# 14 city 3 START')
df_city_3, bem3 = tr_city_2(df_city_3, bem3)
_,__, res3 = learn_step(14, df_city_3, bem3, kimeno)
print('# 14 city 3 END')
if (res1 > res2):
    if(res1 > res3):
        df = df_city_1
        bemenok = bem1
elif (res2 > res3):
    df = df_city_2
    bemenok = bem2
else:
    df = df_city_3
    bemenok = bem3


df_pct_1 = df.copy()
df_pct_2 = df.copy()
bem1 = bemenok.copy()
bem2 = bemenok.copy()
# 15 property_condition_type 1
print('# 15 property_condition_type 1 START')
df_pct_1, bem1 = tr_property_condition_type1(df_pct_1, bem1)
_,__, res1 = learn_step(15, df_pct_1, bem1, kimeno)
print('# 15 property_condition_type 1 END')

# 15 property_condition_type 2
print('# 15 property_condition_type 2 START')
df_pct_2, bem2 = tr_property_condition_type2(df_pct_2, bem2)
_,__, res2 = learn_step(15, df_pct_2, bem2, kimeno)
print('# 15 property_condition_type 2 END')
if (res1 > res2):
    df = df_pct_1
    bemenok = bem1
else:
    df = df_pct_2
    bemenok = bem2

# 16 property_subtype
print('# 16 property_subtype START')
df, bemenok = tr_property_subtype(df, bemenok)
learn_step(16, df, bemenok, kimeno)
print('# 16 property_subtype END')


df_heating_1 = df.copy()
df_heating_2 = df.copy()
bem1 = bemenok.copy()
bem2 = bemenok.copy()
# 17 heating_type 1
print('# 17 heating_type 1 START')
df_heating_1, bem1 = tr_heating_type_1(df_heating_1, bem1)
_,__, res1 = learn_step(17, df_heating_1, bem1, kimeno)
```

```
print('# 17 heating_type 1 END')

# 17 heating_type 2
print('# 17 heating_type 2 START')
df_heating_2, bem2 = tr_heating_type_2(df_heating_2, bem2)
_,__, res2 = learn_step(17, df_heating_2, bem2, kimeno)
print('# 17 heating_type 2 END')
if (res1 > res2):
    df = df_heating_1
    bemenok = bem1
else:
    df = df_heating_2
    bemenok = bem2

# 18 property_floor
df_pfloor_1 = df.copy()
df_pfloor_2 = df.copy()
bem1 = bemenok.copy()
bem2 = bemenok.copy()
# 18 property_floor 1
print('# 18 property_floor 1 START')
df_pfloor_1, bem1 = tr_property_floor_1(df_pfloor_1, bem1)
_,_, res1 = learn_step(18, df_pfloor_1, bem1, kimeno)
print('# 18 property_floor 1 END')

# 18 property_floor 2
print('# 18 property_floor 2 START')
df_pfloor_2, bem2 = tr_property_floor_2(df_pfloor_2, bem2)
_,_, res2 = learn_step(18, df_pfloor_2, bem2, kimeno)
print('# 18 property_floor 2 END')
if (res1 > res2):
    df = df_pfloor_1
    bemenok = bem1
else:
    df = df_pfloor_2
    bemenok = bem2
```

```
# 1 elevator_type START
['elevator_type']
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END ................................, score=-0.326 total time=   2
1.0s
[CV 2/3] END ................................, score=-0.344 total time=   2
2.7s
[CV 3/3] END ................................, score=-0.326 total time=   2
8.5s
# 1 elevator_type END
# 2 county START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END ................................, score=-0.326 total time=   2
1.0s
[CV 2/3] END ................................, score=-0.344 total time=   2
2.7s
[CV 3/3] END ................................, score=-0.326 total time=   2
8.3s
# 2 county END
# 3 room_cnt START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
```

```
[CV 1/3] END ................................., score=-0.325 total time=  2
2.2s
[CV 2/3] END ................................., score=-0.341 total time=  2
4.7s
[CV 3/3] END ................................., score=-0.321 total time=  2
6.4s
# 3 room_cnt END
# 4 view_type START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END ................................., score=-0.321 total time=  2
0.6s
[CV 2/3] END ................................., score=-0.344 total time=  1
9.9s
[CV 3/3] END ................................., score=-0.326 total time=  1
9.1s
# 4 view_type END
# 5 balcony_area START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END ................................., score=-0.312 total time=  5
9.8s
[CV 2/3] END ................................., score=-0.332 total time=  4
0.6s
[CV 3/3] END ................................., score=-0.321 total time=  3
6.0s
# 5 balcony_area END
# 6 garden_access START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END ................................., score=-0.315 total time=  2
3.9s
[CV 2/3] END ................................., score=-0.335 total time=  2
8.3s
[CV 3/3] END ................................., score=-0.322 total time=  2
1.5s
# 6 garden_access END
# 7 created_at => which_day 1 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END ................................., score=-0.315 total time=  2
3.3s
[CV 2/3] END ................................., score=-0.339 total time=  2
3.4s
[CV 3/3] END ................................., score=-0.322 total time=  2
4.1s
# 7 created_at => which_day 1 END
# 7 created_at => which_day 2 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END ................................., score=-0.315 total time=  2
1.9s
[CV 2/3] END ................................., score=-0.339 total time=  2
1.9s
[CV 3/3] END ................................., score=-0.321 total time=  2
2.3s
# 7 created_at => which_day 2 END
# 8 small_room_cnt START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END .................................., score=-0.308 total time=  2
9.0s
[CV 2/3] END .................................., score=-0.332 total time=  2
3.5s
[CV 3/3] END .................................., score=-0.313 total time=  2
1.2s
[CV 2/3] END .................................., score=-0.331 total time=  3
0.8s
[CV 3/3] END .................................., score=-0.314 total time=  4
5.4s
# 9 orientation 1 END
# 9 orientation 2 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END .................................., score=-0.310 total time=  2
3.4s
[CV 2/3] END .................................., score=-0.328 total time=  2
4.0s
[CV 3/3] END .................................., score=-0.315 total time=  2
5.8s
# 9 orientation 2 END
# 10 property_type START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END .................................., score=-0.309 total time=  2
4.4s
[CV 2/3] END .................................., score=-0.331 total time=  3
0.8s
[CV 3/3] END .................................., score=-0.314 total time=  4
5.3s
# 10 property_type END
# 11 postcode 1 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
```

```
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END .................................., score=-0.329 total time= 1.1
min
```

```
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END .................................., score=-0.337 total time= 1.1
min
```

```
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END .................................., score=-0.325 total time= 1.1
min
```

```
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 11 postcode 1 END
# 11 postcode 2 START
```

```
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................, score=-0.329 total time= 1.1
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................, score=-0.337 total time= 1.1
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................, score=-0.325 total time= 1.1
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 11 postcode 2 END
# 12 property_area START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV 1/3] END ................................, score=-0.318 total time= 1.1
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................, score=-0.336 total time= 1.2
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................, score=-0.319 total time= 1.2
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 12 property_area END
# 13 building_floor_count START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................, score=-0.322 total time= 1.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
```

```
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................, score=-0.329 total time= 1.5
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................, score=-0.310 total time= 1.5
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 13 building_floor_count END
# 14 city 1 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................, score=-0.222 total time= 1.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................, score=-0.212 total time= 1.6
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................, score=-0.204 total time= 1.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 14 city 1 END
# 14 city 2 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................, score=-0.309 total time= 1.6
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................, score=-0.310 total time= 1.7
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
```

```
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................., score=-0.300 total time= 1.7
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 14 city 2 END
# 14 city 3 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................., score=-0.309 total time= 2.8
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................., score=-0.310 total time= 2.1
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................., score=-0.300 total time= 1.2
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 14 city 3 END
# 15 property_condition_type 1 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................., score=-0.202 total time= 1.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................., score=-0.191 total time= 1.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................., score=-0.186 total time= 1.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
```

```
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 15 property_condition_type 1 END
# 15 property_condition_type 2 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END .................................., score=-0.215 total time= 1.5
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END .................................., score=-0.207 total time= 1.5
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END .................................., score=-0.199 total time= 1.5
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 15 property_condition_type 2 END
# 16 property_subtype START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END .................................., score=-0.182 total time= 3.0
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END .................................., score=-0.174 total time= 2.5
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END .................................., score=-0.170 total time= 2.6
min
# 16 property_subtype END
# 17 heating_type 1 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................., score=-0.183 total time= 2.8
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................., score=-0.177 total time= 2.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................., score=-0.171 total time= 3.0
min
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................., score=-0.189 total time= 2.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................., score=-0.188 total time= 2.6
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................., score=-0.175 total time= 2.4
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 17 heating_type 2 END
# 18 property_floor 1 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................., score=-0.186 total time= 1.5
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................., score=-0.176 total time= 1.3
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................., score=-0.168 total time= 1.5
min
```

```
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
# 18 property_floor 1 END
# 18 property_floor 2 START
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/3] END ................................, score=-0.177 total time= 1.9
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/3] END ................................, score=-0.177 total time= 1.9
min
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/3] END ................................, score=-0.164 total time= 2.7
min
# 18 property_floor 2 END
/opt/conda/lib/python3.9/site-packages/sklearn/neural_network/_multilayer_per
ceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

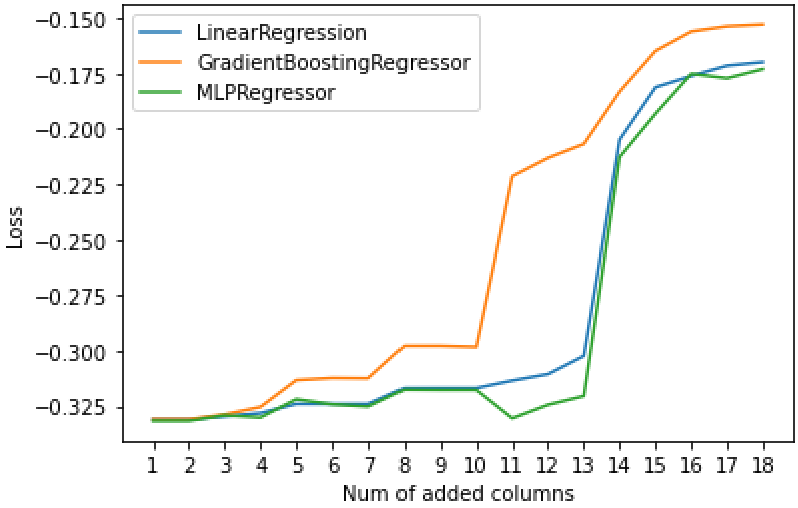
In [12]:

```python
def filter_and_find_largest(data):
    grouped_data = collections.defaultdict(list)

    for item in data:
        first_element = item[0]
        third_element = item[2]
        grouped_data[first_element].append((third_element, item))

    filtered_data = []

    for key, group in grouped_data.items():
        max_element = max(group, key=lambda x: x[0])
        filtered_data.append(max_element[1])

    filtered_data.sort(key=lambda x: x[0])

    return filtered_data
```

In [17]:

```python
score_lin = filter_and_find_largest(score_lin)
score_gbm = filter_and_find_largest(score_gbm)
score_neu = filter_and_find_largest(score_neu)

with open("score_lin.txt", "w") as file:
    file.write(str(score_lin))

with open("score_gbm.txt", "w") as file:
    file.write(str(score_gbm))
```

```
with open("score_neu.txt", "w") as file:
    file.write(str(score_neu))


x_values1 = [point[0] for point in score_lin]
y_values1 = [point[2] for point in score_lin]

x_values2 = [point[0] for point in score_gbm]
y_values2 = [point[2] for point in score_gbm]

x_values3 = [point[0] for point in score_neu]
y_values3 = [point[2] for point in score_neu]

# Create a scatter plot for each dataset
plt.xticks(np.arange(min(x_values1), max(x_values1)+1, 1.0))
plt.plot(x_values1, y_values1, label='LinearRegression', linestyle='-')
plt.plot(x_values2, y_values2, label='GradientBoostingRegressor', linestyle='
plt.plot(x_values3, y_values3, label='MLPRegressor', linestyle='-')

# Label the axes
plt.xlabel('Num of added columns')
plt.ylabel('Loss')

plt.legend()
plt.show()
```



A táblázatban látható a loss-ok alakulása az újabb és újabb oszlopok hozzáadásával, modellenként külön-külön. A paraméterek hozzáadásával általában nő a modellek pontossága. (Kivétel volt ezazlól az MLPRegressor, ami két esetben is gyengébb eredményt hozott új paraméter hozzáadásakor: irányítószámok, fűtéstípus)

A 11. paraméter hozzáadásával a GradientBoostigRegressor nagyot javult, ez az irányítószámok hozzáadása volt. A másik két modell a 14. paraméternél, a kerület hozzáadásakor javult sokat.

A legjobban teljesítő modell a GradientBoostigRegressor lett, de a különbségek nem nagyok. A GradientBoostingRegressor bár differenciálással dolgozik, nincsenek neuronjai, így nem klasszikus neurális hálózat. Érdekes, hogy a neurális hálók nem mindig a legalkalmasabbak minden feladatra, sokszor egy egyszerű lináris regresszióval is megoldhatóak bizonyos feladatok.

Végső adatelemzési folyamat

In [27]:

```
df_final = df.copy()
bemenok_final = bemenok.copy()
```

In [26]:

```
pd.set_option('display.max_columns', None)
```

In [28]:

```
p_grid = {
    'max_depth': [4,5,6,7],
    'n_estimators': [100, 150, 200]
}
g_final = GridSearchCV(
    estimator=GradientBoostingRegressor(random_state=42),
    param_grid=p_grid,
    cv=3, scoring=mape_scorer,
    verbose=3
)
g_final.fit(df_final[bemenok_final], df_final[kimeno])
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
[CV 1/3] END ....max_depth=4, n_estimators=100;, score=-0.165 total time=  1
8.7s
[CV 2/3] END ....max_depth=4, n_estimators=100;, score=-0.161 total time=  1
8.0s
[CV 3/3] END ....max_depth=4, n_estimators=100;, score=-0.152 total time=  1
8.4s
[CV 1/3] END ....max_depth=4, n_estimators=150;, score=-0.158 total time=  2
5.2s
[CV 2/3] END ....max_depth=4, n_estimators=150;, score=-0.154 total time=  2
6.3s
[CV 3/3] END ....max_depth=4, n_estimators=150;, score=-0.146 total time=  2
6.9s
[CV 1/3] END ....max_depth=4, n_estimators=200;, score=-0.155 total time=  3
5.5s
[CV 2/3] END ....max_depth=4, n_estimators=200;, score=-0.150 total time=  3
5.4s
[CV 3/3] END ....max_depth=4, n_estimators=200;, score=-0.143 total time=  3
4.2s
[CV 1/3] END ....max_depth=5, n_estimators=100;, score=-0.160 total time=  2
1.8s
[CV 2/3] END ....max_depth=5, n_estimators=100;, score=-0.154 total time=  2
2.5s
[CV 3/3] END ....max_depth=5, n_estimators=100;, score=-0.146 total time=  2
2.0s
[CV 1/3] END ....max_depth=5, n_estimators=150;, score=-0.154 total time=  3
3.8s
[CV 2/3] END ....max_depth=5, n_estimators=150;, score=-0.147 total time=  3
3.1s
[CV 3/3] END ....max_depth=5, n_estimators=150;, score=-0.141 total time=  3
2.7s
[CV 1/3] END ....max_depth=5, n_estimators=200;, score=-0.150 total time=  4
6.0s
[CV 2/3] END ....max_depth=5, n_estimators=200;, score=-0.144 total time=  4
4.4s
[CV 3/3] END ....max_depth=5, n_estimators=200;, score=-0.138 total time=  3
9.9s
[CV 1/3] END ....max_depth=6, n_estimators=100;, score=-0.154 total time=  2
3.5s
[CV 2/3] END ....max_depth=6, n_estimators=100;, score=-0.148 total time=  2
2.8s
[CV 3/3] END ....max_depth=6, n_estimators=100;, score=-0.142 total time=  2
2.7s
```

```
[CV 1/3] END ....max_depth=6, n_estimators=150;, score=-0.150 total time=  3
3.7s
[CV 2/3] END ....max_depth=6, n_estimators=150;, score=-0.143 total time=  3
4.0s
[CV 3/3] END ....max_depth=6, n_estimators=150;, score=-0.137 total time=  3
3.8s
[CV 1/3] END ....max_depth=6, n_estimators=200;, score=-0.147 total time=  4
6.2s
[CV 2/3] END ....max_depth=6, n_estimators=200;, score=-0.140 total time=  4
6.5s
[CV 3/3] END ....max_depth=6, n_estimators=200;, score=-0.136 total time=  4
4.9s
[CV 1/3] END ....max_depth=7, n_estimators=100;, score=-0.152 total time=  2
6.3s
[CV 2/3] END ....max_depth=7, n_estimators=100;, score=-0.144 total time=  2
6.6s
[CV 3/3] END ....max_depth=7, n_estimators=100;, score=-0.139 total time=  2
6.3s
[CV 1/3] END ....max_depth=7, n_estimators=150;, score=-0.148 total time=  4
5.5s
[CV 2/3] END ....max_depth=7, n_estimators=150;, score=-0.140 total time=  4
6.7s
[CV 3/3] END ....max_depth=7, n_estimators=150;, score=-0.135 total time=  4
3.1s
[CV 1/3] END ....max_depth=7, n_estimators=200;, score=-0.147 total time=  5
2.0s
[CV 2/3] END ....max_depth=7, n_estimators=200;, score=-0.138 total time=  5
2.3s
[CV 3/3] END ....max_depth=7, n_estimators=200;, score=-0.134 total time=  5
2.6s
```

Out[28]:

```
▸              GridSearchCV
▸ estimator: GradientBoostingRegressor
        ▸ GradientBoostingRegressor
```

In [39]:

```
best = g_final.best_estimator_
type(best)
g_final.best_estimator_
```

Out[39]:

```
▾                    GradientBoostingRegressor
GradientBoostingRegressor(max_depth=7, n_estimators=200, random_state
=42)
```

A legjobb modell a vizsgáltak közül: GradientBoostingRegressor(max_depth=7, n_estimators=200, random_state=42)

In [40]:

```
def kiertekelo_fuggveny(df, tipp_oszlop_neve, target_oszlop_neve):
    mean_absolute_error=   (abs(df[tipp_oszlop_neve]-df[target_oszlop_neve]))

    MAPE=(abs((df[target_oszlop_neve] - df[tipp_oszlop_neve])/df[target_oszlo
    rmse =  np.sqrt( (  (df[tipp_oszlop_neve]-df[target_oszlop_neve])**2 ).mea
    print("RMSE",rmse)
    print("MAE:",mean_absolute_error)
    print("MAPE:",MAPE)
    df["tipp_arany"] = df[target_oszlop_neve]/df[tipp_oszlop_neve]
    erdekesek = df[ (df['tipp_arany']> 0.75) & (df['tipp_arany'] < 0.95)]
    print("Atnezendok aranya", len(erdekesek)/len(df))
    return erdekesek
```

In [45]:

```
# A feladat egy olyan gépi tanulási megoldás elkészítése, ami egy ingatlan ad

test_df = df.copy()
test_df['tipp']=best.predict(test_df[bemenok_final])
erdekesek = kiertekelo_fuggveny(test_df, 'tipp', 'jelen_nmAr')

df_to_write = pd.read_csv('data/DataSet_LakasArak.csv')
df_to_write = df_to_write.filter(items=erdekesek.index, axis=0)
df_to_write.to_csv('Results_O78UXU.csv', index=False)
```

```
RMSE 84395.73240336076
MAE: 55183.947416220566
MAPE: 0.115254033852456
Atnezendok aranya 0.33756477673512525
```