ML-1 Challenge1
House Price Prediction - Inspired from Russian Housing

FEBRUARY 23, 2022
MUHAMMAD AMIN GHIAS -25366

Table of Contents

ABSTRACT:

Buying new house is a difficult task, this project will predict the price of houses using the given dataset. This will help people in both buying and selling houses. People can select their specific features they need for a house they want to buy and get its appropriate predictive price respectively.

INTRODUCTION

People in this fast paced and competitive world have to move and change homes. Changing homes within a big city takes a lot of effort and time and becomes a huge task which has severe consequences on their life. This project will help people in their decision of buying or selling a home by predicting the price of their required features house

BUSINESS PROBLEM & MAIN OBJECTIVE

Everybody has their own needs requirements when it comes to making a decision for buying a house. Shifting home in a large city is quite a challenging task and requires a wise decision. Many factors are to be taken in consideration when one shifts home.

The problem people shifting home are the availability of following in the new area:

- Restaurants
- Shopping centers
- Grocery Stores
- Parks
- Cinemas
- Transport buses

With the more features and facilities, the price will vary accordingly. Hence in this project we will predict the prices with the given/desired features.

DATA ACQUISITION AND WRANGLING

DATA SOURCE

For this project we will use online data the source is from kaggle

with url:" https://www.kaggle.com/c/ml-1-challenge1/data"

The dataset used in the competition is inspired by the original Russian Housing Price Prediction dataset hosted on Kaggle. However, through SMOTE and Stresser, the actual data has been changed significantly. The revised(synthesized) dataset has around 260K rows whereas the original dataset had 30K rows.

2 different dataset are given, 1 train dataset and 1 test dataset is provided

The train dataset consists of 272 columns and 181507 rows

DATA DESCRIPTION

A brief description of dataframe is as:

```
df=pd.read_csv('train.csv')
```

```
df.shape
✓ 0.1s
(181507, 272)
```

```
df.info()
✓ 0.2s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181507 entries, 0 to 181506
Columns: 272 entries, full_sq to price_doc
dtypes: float64(257), object(15)
memory usage: 376.7+ MB
```

The train dataset consists of of 272 columns and 181507 rows

The 272 columns have 257 floats and 15 objects values

```
df.head()
✓ 0.2s
```

	full_sq	life_sq	floor	product_type	sub_area	area_m	raion_popul	gree	count_5000	sport_count_5000	market_count_5000	price_doc
0	43.0	27.0	4.0	Investment	Bibirevo	6407578.100	155572.0		0.0	52.0	4.0	5850000.0
1	34.0	19.0	3.0	Investment	Nagatinskij Zaton	9589336.912	115352.0		10.0	66.0	14.0	6000000.0
2	43.0	29.0	2.0	Investment	Tekstil'shhiki	4808269.831	101708.0		4.0	67.0	10.0	5700000.0
3	77.0	77.0	4.0	Investment	Basmannoe	8398460.622	108171.0		91.0	195.0	14.0	16331452.0
4	67.0	46.0	14.0	Investment	Nizhegorodskoe	7506452.020	43795.0		20.0	113.0	17.0	9100000.0

5 rows × 272 columns

Out of 272 columns there are 271 features and 1 target feature ‘price_doc’

As our ‘price_doc’ target is of floats value so our prediction problem is of Regression Nature:

DATA CLEANING

Formulating dataset into all float continuous values:

For our analysis to proceed further we need all features 271 of them as same data type that is all must be either floats or integers, Our dataframe has 257 features as float and 15 object type features.

These 15 object types, which have many classes and their unique count are as follows

```
[10]  df_obj=dataset_objs(df)

...   product_type : ['product_type', array(['Investment', 'OwnerOccupier'], dtype=object), 2] count : 2
      sub_area : ['sub_area', array(['Bibirevo', 'Nagatinskij Zaton', "Tekstil'shhiki", ...,
             'Poselenie iMhajlovo-Jarcevskoe', 'Poselenie Mihajlovo-Jracevskoe',
             'Poselenie Mhiajlovo-Jarcevskoe'], dtype=object), 1927] count : 1927
      culture_objects_top_25 : ['culture_objects_top_25', array(['no', 'yes'], dtype=object), 2] count : 2
      thermal_power_plant_raion : ['thermal_power_plant_raion', array(['no', 'yes'], dtype=object), 2] count : 2
      incineration_raion : ['incineration_raion', array(['no', 'yes'], dtype=object), 2] count : 2
      oil_chemistry_raion : ['oil_chemistry_raion', array(['no', 'yes'], dtype=object), 2] count : 2
      radiation_raion : ['radiation_raion', array(['no', 'yes'], dtype=object), 2] count : 2
      railroad_terminal_raion : ['railroad_terminal_raion', array(['no', 'yes'], dtype=object), 2] count : 2
      big_market_raion : ['big_market_raion', array(['no', 'yes'], dtype=object), 2] count : 2
      nuclear_reactor_raion : ['nuclear_reactor_raion', array(['no', 'yes'], dtype=object), 2] count : 2
      detention_facility_raion : ['detention_facility_raion', array(['no', 'yes'], dtype=object), 2] count : 2
      water_1line : ['water_1line', array(['no', 'yes'], dtype=object), 2] count : 2
      big_road1_1line : ['big_road1_1line', array(['no', 'yes'], dtype=object), 2] count : 2
      railroad_1line : ['railroad_1line', array(['no', 'yes'], dtype=object), 2] count : 2
      ecology : ['ecology', array(['good', 'excellent', 'poor', 'satisfactory', 'no data'],
            dtype=object), 5] count : 5
```

The results above shows that sub_area has 1927 unique classes

For this purpose, we required data cleaning and remove objects or convert their data types. In this regard we took 3 approaches and made 3 types of dataframes for our initial models. These 3 approaches were:

1. Remove all the 15 objects from our dataframe and do our analysis on it
2. Convert datatype of all objects into dummy variables using one hot encoding
3. Removing specific objects ('sub_area') while keeping other and apply one hot encoding on them.

We started our analysis using all 3 approaches to find which dataset will give best prediction results on normal Linear regression

First Approach:

Removing 15 object features

```
df_rem=df.copy()
df_rem.head()

df_rem= df_rem.drop(columns=col_objrem)
df_rem.head()
```

	full_sq	life_sq	floor	area_m	raion_popul	g
0	43.0	27.0	4.0	6407578.100	155572.0	
1	34.0	19.0	3.0	9589336.912	115352.0	
2	43.0	29.0	2.0	4808269.831	101708.0	
3	77.0	77.0	4.0	8398460.622	108171.0	
4	67.0	46.0	14.0	7506452.020	43795.0	

5 rows × 257 columns

Using this dataset we found Linear Regression (LR) with and without filtering which gave better results

Second Approach:

Converting full dataset using one hot encoding made dataset with:

```
adf.shape
```

```
(181507, 2214)
```

Using this dataset we found Linear Regression (LR) without filtering which gave worse results

Third Approach

Removing on ‘sub_area’ feature from dataset and applying one hot encoding and removing ‘price_doc’

This will give 287 rows

```
Applying onehot encoding to convert
```

```
X = pd.get_dummies(dfX)
print(X.shape)
X.head()
```

```
(181507, 287)
```

Using this dataset we found Linear Regression (LR) with and without filtering which gave best results

Summary of the 3 approaches is shown below in table

Approach	Description	LR results MSE
1	Dataset without 15 objects	13302113.55
2	Full dataset	13310892.13
3	Dataset without 'sub_area' object	13279234.85

The above results show that approach 3 is best to start our prediction analysis

The above results show that 'sub_area' has very less effect on predicting the price and can be neglected whereas the other 14 object features have more relation to price and keeping them gives better price prediction.

With the help of above observation we have concluded that approach 3 is best and we will keep the dataset used in approach 3 as our datframe for our analysis.

FILTERING:

In our analysis:

X = train datframe

y = target 'price_doc'

Xt = train dataframe

Before applying filtering we found the correlation matrix of our dataframe X

FEATURE REMOVAL ON BASIS OF CORRELATION

Using Correlation we did filtering on dataset in two ways:

1. **Removing features on basis of self correlation (high correlation values) of features**

```
rem_selfcor =self_corr(cor,0.8,1)
✓ 1.6s
```

	feature 1	feature 2	correlation values
0	office_count_5000	cafe_count_5000	0.816991
1	office_count_5000	cafe_count_5000_na_price	0.811909
2	office_count_5000	cafe_count_5000_price_500	0.815336
3	office_count_5000	cafe_count_5000_price_1000	0.816065
4	office_count_5000	cafe_count_5000_price_1500	0.812276

As features with high self correlation are nearly equivalent therefore one of them can be neglected. Features with high correlation from 0.8 to 1 correlations were found and all feature 1 were removed and feature 2 were left in dataframe X

2. **Removing features on basis of correlation with y ('price_doc)**

Feature with correlation with 'price_doc' less than 0.4 removed from dataframe X

```
rem_cor=target_cor(cor,'price_doc',0.4)
✓ 0.1s
raion_popul                          0.255183
green_zone_part                      0.268517
indust_part                          0.253134
children_preschool                   0.368395
preschool_education_centers_raion    0.290759
                                        ...
cafe_sum_5000_max_price_avg          0.359831
cafe_avg_price_5000                  0.366213
mosque_count_5000                    0.270513
sport_count_5000                     0.300094
market_count_5000                    0.236708
Name: price_doc, Length: 79, dtype: float64
```

79 features removed

FEATURE REMOVAL ON BASIS OF VARIANCE

Now we can also remove features whose values remains constant that is their value does not change and is nearly similar, so it does not participate in predicting the price

```
rem_var=self_var(X,0.1)
✓ 1.1s
green_zone_part                0.045573
indust_part                    0.019900
mosque_count_500               0.047256
mosque_count_1000              0.056570
mosque_count_1500              0.071232
mosque_count_2000              0.096186
incineration_raion_no          0.088340
incineration_raion_yes         0.088340
oil_chemistry_raion_no         0.079872
oil_chemistry_raion_yes        0.079872
nuclear_reactor_raion_no       0.093696
nuclear_reactor_raion_yes      0.093696
big_road1_1line_no             0.099714
big_road1_1line_yes            0.099714
railroad_1line_no              0.096000
railroad_1line_yes             0.096000
dtype: float64
```

The above features were removed from X dataframe

FEATURE REMOVAL ON P-VALUE BASIS

We also removed features whose p-value is greater than 0.05

```
rem_p=p_val(X)
✓ 13.4s

Output exceeds the size limit. Open the full output
(181507, 287)
(181507, 287)
area_m : 0.48714782552338587
indust_part : 0.7380758309531251
school_education_centers_raion : 0.15144324123398153
sport_objects_raion : 0.08615826192782934
culture_objects_top_25_raion : 0.9533386797348384
full_all : 0.9737798695484825
young_all : 0.5105806323639629
young_male : 0.4160941168152975
work_all : 0.3666113017495538
ekder_all : 0.09255879403431207
0_6_all : 0.9998129717828271
0_6_male : 0.4328441569562722
7_14_all : 0.6813597176856977
7_14_male : 0.9331362843786709
7_14_female : 0.11001012209248583
0_17_all : 0.06024453118050101
```

After filtering datframe and removing feature we rename it to X2

X2= filtered dataframe

Finding best filtering parameters

For choosing the required best parameters for filtering for LR we used different combinations of filtering was done to chose which gives best results

Some variations with p values>0.05 were

1) Feature removal on 0.75 self-correlation, 0.4 price correlation and variance 0.2. bad MSE result
2) Feature removal on 0.75 self-correlation, 0.25 price correlation and variance 0.1. bad MSE result
3) Feature removal on 0.8 self-correlation, 0.25 price correlation and variance 0.1. bad MSE result
4) Feature removal on 0.8 self-correlation, 0.4 price correlation and variance 0.1. best MSE result

So choice 4 was chosen that is removing features from X :

1) With self-correlation >0.8 and <1
2) With correlation with price <0.4
3) Variance <0.1
4) P values >0.05

Similarly the test dataset filtered

Xt = test datframe

X2t = filtered test dataframe

DATA ANALYSIS & USE OF MODELS:

NORMAL LINEAR REGRESSION:

Normal Linear Regression was applied on X and X2 with and without using filtering

	LR results MSE	
Description	**Without filtering**	**With filtering**
Dataset without 'sub_area' object	13279234.85	13316918.55

The results showed that doing filtering did not reduce our MSE using LR

SFS FORWARD FEATURE SELECTION

The SFS forward feature selection were done with following results:

1. 15 features
2. 20 features **best R2 results**
3. 25 features
4. 30 features

The 20 features selected by forward SFS

```
[99]  f_20=f_sel(X2,20)

...  ['full_sq' 'life_sq' 'floor' 'school_education_centers_top_20_raion'
      'male_f' 'build_count_monolith' 'green_zone_km' 'industrial_km'
      'church_synagogue_km' 'prom_part_500' 'trc_sqm_500'
      'cafe_count_500_price_high' 'leisure_count_500' 'green_part_1000'
      'office_sqm_1000' 'cafe_count_1000_price_high' 'leisure_count_1000'
      'market_count_1000' 'trc_sqm_2000' 'sport_count_2000']
     0.6139179922467107
```

POLYNOMIAL FEATURES

Now we made polynomial of degree 2 on the 20 selected features

```
X20p = poly_df(X20)
```

```
Output exceeds the size limit. Open the full output data in a text editor
['1' 'full_sq' 'life_sq' 'floor' 'school_education_centers_top_20_raion'
 'male_f' 'build_count_monolith' 'green_zone_km' 'industrial_km'
 'church_synagogue_km' 'prom_part_500' 'trc_sqm_500'
 'cafe_count_500_price_high' 'leisure_count_500' 'green_part_1000'
 'office_sqm_1000' 'cafe_count_1000_price_high' 'leisure_count_1000'
 'market_count_1000' 'trc_sqm_2000' 'sport_count_2000' 'full_sq^2'
 'full_sq life_sq' 'full_sq floor'
 'full_sq school_education_centers_top_20_raion' 'full_sq male_f'
 'full_sq build_count_monolith' 'full_sq green_zone_km'
 'full_sq industrial_km' 'full_sq church_synagogue_km'
 'full_sq prom_part_500' 'full_sq trc_sqm_500'
 'full_sq cafe_count_500_price_high' 'full_sq leisure_count_500'
```

Using polynomial features, we made 231 features we made X20p dataframe

On applying LR on this dataset of 20 feature selected and using polynomial we got the best result of LR with MSE 13152092.20746

KNN K NEAREST NEIGHBOUR:

For using KNN datset was scaled using Min Max Scaling

We applied KNN with K=90, 100, 120

Also used cross validation to find best value of K

```
Iter: 70, Mean-KNN R2: 0.6277, Mean-LR : R2: 0.6216, Change: 4.6156
Iter: 80, Mean-KNN R2: 0.6264, Mean-LR : R2: 0.6216, Change: -0.2030
Iter: 90, Mean-KNN R2: 0.6253, Mean-LR : R2: 0.6216, Change: -0.1801
Iter: 100, Mean-KNN R2: 0.6241, Mean-LR : R2: 0.6216, Change: -0.1860
Iter: 110, Mean-KNN R2: 0.6230, Mean-LR : R2: 0.6216, Change: -0.1857
Iter: 120, Mean-KNN R2: 0.6216, Mean-LR : R2: 0.6216, Change: -0.2131
Iter: 130, Mean-KNN R2: 0.6204, Mean-LR : R2: 0.6216, Change: -0.2048
Iter: 140, Mean-KNN R2: 0.6193, Mean-LR : R2: 0.6216, Change: -0.1733
```

Using KNN we got much better results than LR

Applied KNN on X, X2 and that and we got following results

KNN			
S.no	**Data Set**	**K**	**RMSE**
1	X no filtering	100	12861389.91
2	X2 with filtering	100	12868096.1
3	X2 with filtering	90	12865066.75

This shows that KNN showed with full dataframe X without filtering shows best result indicating that features with filter are important for predicting price.

RANDOM FORREST REGRESSOR/CLASSISFIER

Finally, regression tree model Random Forrest regressor model was used for X and X20 with following results:

Random Forrest			
S.no	**Data Set**	**Trees**	**RMSE**
1	X no filtering	100	12499675.32
2	X2 with filtering	100	12514244.33

This shows that for X we get the best result and least RMSE

Finding optimal number of tress/ estimatator

We tried to find the best optimal number of tress to give least error, it was analyzed on reduced rows 11507 (small portion of X2) rows of X2 which gave us the following results

n_trees	oob
15.0	0.970803
20.0	0.968812
30.0	0.967485
40.0	0.964831
50.0	0.965494
100.0	0.961513
150.0	0.964167
200.0	0.967485
300.0	0.969476
400.0	0.971466

Here OOB is the error which shows at 100 trees we get least error, hence 100 trees is better choice.

DISCUSSION:

We chose which dataset gave best results out of our 3 approaches. On the selected dataset we Applied Normal Linear Regression, KNN and Random Forrest with and without filtering.

RESULTS:

After applying different methods we got following summarized results:

		LR results MSE	
S.No	**Description**	**Without filtering**	**With filtering**
1	Dataset without 'sub_area' object	13279234.85	13316918.55

KNN			
S.no	**Data Set**	**K**	**RMSE**
1	X no filtering	100	12861389.91
2	X2 with filtering	100	12868096.1
3	X2 with filtering	90	12865066.75

Random Forrest			
S.no	**Data Set**	**Trees**	**RMSE**
1	X no filtering	100	12499675.32
2	X2 with filtering	100	12514244.33

CONCLUSION

Following major observations concluded:

- After our various attempts we found that ‘sub_area’ column feature had minimal effect on our perdition of price and it was removed .
- Best result of LR were achieved by using 2 degree polynomial on 20 features selected this shows that out data is non linear
- Using filtering techniques on all of our model did not improve results and increased RMSE, this shows that the features removed by filtering had a significant effect on predicting the price
- Of all 3 models random forrest gives least RMSE and best results

SUGGESTIONS

Some suggestions for future work are:

- Using gridsearch and finding optimal number of trees for random forrest we can improve results and reduce error
- The removed feature ‘sub_area’ had 1927 unique values a few values with high frequency can be added in dataset (that is top 15 most repeated sub_area classes added in dataset) addition of 15 features may improve the results.