Capstone Project
Predicting Real Estate Prices In Moscow

Domain Background

Moscow's housing market flourished in the last decade, setting and outstripping multiple records. In 2015 3.8 million square metres of new housings were built. This is four times as much as were build in New York City in 2014. This upward trend is in sharp contrast to Russia's overall economic situation. Using machine learning techniques to forecast housing prices has been done and is done by various property agents (e.g.foxtons [1]) and search websites [2]. However, how exactly the employed algorithms work, which are employed is not directly stated. There is however some literature found describing attempts of housing price predictions. One is applying various regression techniques in order to determine housing prices in London [3]. Here, gaussian processes, which define a distribution over parameters, performed best, still worse than websites which served as benchmark. The main reason for this was stated to be the limited amount of data available. In a another study the authors employed different machine learning algorithms to predict housing prices in Fairfax County, Virginia. A 10-fold cross validation was applied to C4.5 decision tree, Bayesian, Adaboost and RIPPER (Repeated Incremental Pruning to Produce Error Reduction). Of those algorithms RIPPER and Adaboost performed better on predicting housing prices than C4.5 or Bayesian [4]. Also a couple of papers describe the use of artificial neuronal networks in order to predict housing prices [5,6]. Further, tree based algorithms [7] and combinations of genetic algorithms and support vector machines [8] were used in order to predict housing prices.

Problem Statement

The country has a volatile economic showing multiple up and downs and no clear trend. Since housing prices are directly influenced by the country's economic, this makes it hard to predict housing prices. Investment in properties is a huge expense, for private investors as well as for companies, therefore a reliable prediction of possible expenses is needed. Real life data is often messy, error prone and showing a high degree of colinearity, which makes feature selection and engineering a very important step of the machine learning process. The target variable of the data set is the housing price which is a continuous numerical variable. Therefore, I will attempt a regression in order to predict the housing prices. The biggest problem will be to preprocess the data in order to meet the requirements for linear regression, namely: a linear relationship between features and target variable, multivariate normality, no or little multicollinearity, no auto-correlation and homoscedasticity. A further problem might be the interpretation of the results. Besides the predicted values, it would be nice to gain insights into how strong the individual features impact the housing prices.

Dataset and inputs

The data I will use for this capstone project is provided by the Sberbank Russia as part of a Kaggle challenge [9]. The data comes in three separate files, namely a training and a testing csv file and a csv file with various data regarding Russia's economics (train.csv, test.csv and macro.csv, respectively). The training dataset has 30,471 entries with 291 features and one target variable (sale price). The testing dataset has 7,662 entries with 291 features. The macro.csv file has 2484 entries with 100 features. The macro.csv file as well as the test.csv and train.csv all share a timestamp feature. The test.csv and train.csv files are mostly composed of discrete and continuous numeric values (157 features of integer type and 119 features of float type) and only some nominal (e.g. sub_area) and ordinal (e.g. ecology) features (16 features). The macro.csv file is composed of 96 continuous numeric values (2 integer and 94 float types) and 4 features displayed as object type. The macro includes features of importance for the Russian economic. For instance the country's GDP, the exchange rate of rubel toward other currencies like euro. Further, information about the general population is provided like mortality, childbirth or number of marriages. Train.csv and test.csv contain features about the specific houses. Important features, for instance, are square meter, living room m², build_year, population density of the neighbourhood. Besides property specific information also features regarding the neighbourhood are provided. Those are for instance number of schools, shopping centers or cafes in the neighbourhood. Also some information of the location of the property can be found in the features. For example in features like kremlin_km, railroud_1line etc.

Solution statement

After cleaning the data, imputing features and feature reduction a regression model will be applied in order to predict the housing prices in Moscow. The analysis of the reduced feature set will allow interpretation of features which are primarily determining housing prices in Moscow. Using linear regression imposes some requirements on the data which should be met, I will attempt a ridge regression, using grid search to determine a valid alpha value will help reduce overfitting of the model. On the other hand I will also apply the XGBoost algorithm. XGBoost is one of the most successful gradient boosting algorithms. It basically is an ensemble learner based on trees. Those trees are shallow, therefore weak learners. A tree-based regression will also grant me insight into the importance of the features.

Benchmark Model

In order to benchmark my solution I am going to use the regression algorithm on the full feature set which should lead to a decreased predictive power of the model, according to Hughes phenomenon [10]. Also, the out of the box machine learning algorithms will be used to show if parameter tuning has a positive effect on the predictive power of the algorithms. On the other hand, I will be able to measure my model's performance against other Kaggler model's performance. I will attempt to reach the top 50% in this competition.

Evaluation metrics

The Root Mean Squared Logarithmic Error (RMSLE) will be used in combination with a 10-fold cross validation (CV) as the evaluation metric. The RMSLE will penalize (big) differences between big numbers less than would the Root Mean Squared Error. For the CV the data will be split in a 80:20 ratio (test,train*, respectively).

The RMSLE is calculated the following [11]:

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

ϵ: RMSLE value (score)

n : total number of observations in the (public/private) data set
pi : prediction
ai : actual response for *i*
log(*x*) : natural logarithm of *x*

Project design
Before starting to actually work with the data I will merge the train.csv (and test.csv) with the macro.csv via their time-stamp feature. After this, I will start with an throughout exploration of the dataset. Since, the dataset is large and noisy I will try to clean it before making any predictions. Therefore, I will employ graphical representation and descriptive statistics to identify rows with a lot of missing feature values and features without information (i.e. zero variance). Features with low variance will be removed, as well as features with too much missing values. Other missing values are going to be imputed, in a first attempt i will use the features median to impute the missing entries. Also features which are of no clear use for the predictive model will be removed (e.g. id_features). In the second step of the preprocessing, I will try to detect outliers and erroneous data points. For the outlier detection, I will attempt to use an isolation forest. The detected outliers will be removed or, if possible changed into sensible values. Erroneous data points (e.g. total area of property < kitchen area) will be removed or changed (e.g. swap kitchen area with total area (if reasonable)). Further, object data of the data set will be hot encoded in order to be useful for the predictive model. In the third step of data preprocessing I will try to identify important features of the dataset. For this I will apply a RandomForestRegressor, which is good since it requires little feature engineering and will show feature importance. Afterwards, I will try to engineer new features based on combinations of the most important ones.
In the fourth step of the data preprocessing I will look at the distribution of the data. Here, I will use barplots as well as statistical tests to look at the distribution of the data. Python's scipy module provides some useful methods for this (e.g. *scipy.stats.mstats.normaltest* for normality). Non-normal distributed data will be transformed depending on their distribution (e.g. log transformation for skewed data). Finally, the data will be scaled in order to display zero mean and unit variance.
The second part will be to further reduce the number of features of the dataset. Here, I will look at correlation of the features via a correlation matrix. After accessing the correlation of the dataset, I will remove features leading to high multicollinearity. Therefore, I will calculate the variance inflation factor (VIF) for the features. As a rule of thumb a VIF > 10 indicates collinearity in the dataset. Therefore, as long as this threshold is exceeded I will drop the feature with the highest VIF.
The third part will be the training of a regression model. I will try and compare two different regressors on this data set. On the one hand I will use a ridge regression model and on the other hand a gradient boosting regressor (xgboost [12]). For the alpha parameter of the ridge regression model as well as for some of the xgboost parameters a grid search will be performed. The train.csv dataset will be split into a training and testing part and afterwards the models will be fitted on the training and tested on the testing part. To ensure consistency and reliability of prediction a 10-fold cross validation will be performed. The goodness of the models in terms of train and test accuracy will be graphically represented.

The fourth and final part of the project will be the interpretation of the results. I will attempt to answer question like what are the primary determinants of housing prices and how is the economy influencing the prices.

[1] https://www.foxtons.co.uk/
[2] http://www.zoopla.co.uk/property/estimate/about/
[3] Aaron Ng, Marc Deisenroth, Machine Learning for a London Housing Price Prediction Mobile Application
[4]Byeonghwa Park, Jae Kwon Bae, Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data, Expert Systems with Applications, Volume 42, Issue 6, 15 April 2015, Pages 2928-2934, ISSN 0957-4174
[5]Núñez Tabales, Julia M.; Caridad y Ocerin, José María; Rey Carmona, Francisco J. (2013) : Artificial neural networks for predicting real estate prices, Revista de Métodos Cuantitativos para la Economía y la Empresa, ISSN 1886-516X, Vol. 15, pp. 29-44
[6]Liu JG., Zhang XL., Wu WP. (2006) Application of Fuzzy Neural Network for Real Estate Prediction. In: Wang J., Yi Z., Zurada J.M., Lu BL., Yin H. (eds) Advances in Neural Networks - ISNN 2006. ISNN 2006. Lecture Notes in Computer Science, vol 3973. Springer, Berlin, Heidelberg
[7]Gang-Zhi Fan, Seow Eng Ong, Hian Chye Koh, Determinants of House Price: A Decision Tree Approach
[8]Jirong Gu, Mingcang Zhu, Liuguangyan Jiang, Housing price forecasting based on genetic algorithm and support vector machine, Expert Systems with Applications, Volume 38, Issue 4, April 2011, Pages 3383-3386, ISSN 0957-4174,
[9] https://www.kaggle.com/c/sberbank-russian-housing-market/kernels
[10] G. Hughes, "On the mean accuracy of statistical pattern recognizers," in *IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55-63, January 1968.
doi: 10.1109/TIT.1968.1054102
[11] https://www.kaggle.com/wiki/RootMeanSquaredLogarithmicError
[12] https://xgboost.readthedocs.io/en/latest/

Macro.csv

timestamp : object
oil_urals : float64
gdp_quart : float64
gdp_quart_growth : float64
cpi : float64
ppi : float64
gdp_deflator : float64
balance_trade : float64
balance_trade_growth : float64
usdrub : float64
eurrub : float64
brent : float64
net_capital_export : float64
gdp_annual : float64
gdp_annual_growth : float64
average_provision_of_build_contract : float64
average_provision_of_build_contract_moscow : float64
rts : float64
micex : float64
micex_rgbi_tr : float64
micex_cbi_tr : float64
deposits_value : int64
deposits_growth : float64
deposits_rate : float64
mortgage_value : int64
mortgage_growth : float64
mortgage_rate : float64
grp : float64
grp_growth : float64
income_per_cap : float64
real_dispos_income_per_cap_growth : float64
salary : float64
salary_growth : float64
fixed_basket : float64
retail_trade_turnover : float64
retail_trade_turnover_per_cap : float64
retail_trade_turnover_growth : float64
labor_force : float64
unemployment : float64
employment : float64
invest_fixed_capital_per_cap : float64
invest_fixed_assets : float64
profitable_enterpr_share : float64
unprofitable_enterpr_share : float64
share_own_revenues : float64
overdue_wages_per_cap : float64
fin_res_per_cap : float64
marriages_per_1000_cap : float64
divorce_rate : float64
construction_value : float64
invest_fixed_assets_phys : float64
pop_natural_increase : float64

pop_migration : float64
pop_total_inc : float64
childbirth : float64
mortality : float64
housing_fund_sqm : float64
lodging_sqm_per_cap : float64
water_pipes_share : float64
baths_share : float64
sewerage_share : float64
gas_share : float64
hot_water_share : float64
electric_stove_share : float64
heating_share : float64
old_house_share : float64
average_life_exp : float64
infant_mortarity_per_1000_cap : float64
perinatal_mort_per_1000_cap : float64
incidence_population : float64
rent_price_4+room_bus : float64
rent_price_3room_bus : float64
rent_price_2room_bus : float64
rent_price_1room_bus : float64
rent_price_3room_eco : float64
rent_price_2room_eco : float64
rent_price_1room_eco : float64
load_of_teachers_preschool_per_teacher : float64
child_on_acc_pre_school : object
load_of_teachers_school_per_teacher : float64
students_state_oneshift : float64
modern_education_share : object
old_education_build_share : object
provision_doctors : float64
provision_nurse : float64
load_on_doctors : float64
power_clinics : float64
hospital_beds_available_per_cap : float64
hospital_bed_occupancy_per_year : float64
provision_retail_space_sqm : float64
provision_retail_space_modern_sqm : float64
turnover_catering_per_cap : float64
theaters_viewers_per_1000_cap : float64
seats_theather_rfmin_per_100000_cap : float64
museum_visitis_per_100_cap : float64
bandwidth_sports : float64
population_reg_sports_share : float64
students_reg_sports_share : float64
apartment_build : float64
apartment_fund_sqm : float64

Train.csv and Test.csv (*price_doc* only in train.csv)

id : int64
timestamp : object
full_sq : int64
life_sq : float64
floor : float64
max_floor : float64
material : float64
build_year : float64
num_room : float64
kitch_sq : float64

state : float64
product_type : object
sub_area : object
area_m : float64
raion_popul : int64
green_zone_part : float64
indust_part : float64
children_preschool : int64
preschool_quota : float64
preschool_education_centers_raion : int64
children_school : int64
school_quota : float64
school_education_centers_raion : int64
school_education_centers_top_20_raion : int64
hospital_beds_raion : float64
healthcare_centers_raion : int64
university_top_20_raion : int64
sport_objects_raion : int64
additional_education_raion : int64
culture_objects_top_25 : object
culture_objects_top_25_raion : int64
shopping_centers_raion : int64
office_raion : int64
thermal_power_plant_raion : object
incineration_raion : object
oil_chemistry_raion : object
radiation_raion : object
railroad_terminal_raion : object
big_market_raion : object
nuclear_reactor_raion : object
detention_facility_raion : object
full_all : int64
male_f : int64
female_f : int64
young_all : int64
young_male : int64
young_female : int64
work_all : int64
work_male : int64
work_female : int64
ekder_all : int64
ekder_male : int64
ekder_female : int64
0_6_all : int64
0_6_male : int64
0_6_female : int64
7_14_all : int64
7_14_male : int64
7_14_female : int64
0_17_all : int64
0_17_male : int64
0_17_female : int64
16_29_all : int64
16_29_male : int64
16_29_female : int64
0_13_all : int64
0_13_male : int64
0_13_female : int64
raion_build_count_with_material_info : float64
build_count_block : float64
build_count_wood : float64
build_count_frame : float64

build_count_brick : float64
build_count_monolith : float64
build_count_panel : float64
build_count_foam : float64
build_count_slag : float64
build_count_mix : float64
raion_build_count_with_builddate_info : float64
build_count_before_1920 : float64
build_count_1921-1945 : float64
build_count_1946-1970 : float64
build_count_1971-1995 : float64
build_count_after_1995 : float64
ID_metro : int64
metro_min_avto : float64
metro_km_avto : float64
metro_min_walk : float64
metro_km_walk : float64
kindergarten_km : float64
school_km : float64
park_km : float64
green_zone_km : float64
industrial_km : float64
water_treatment_km : float64
cemetery_km : float64
incineration_km : float64
railroad_station_walk_km : float64
railroad_station_walk_min : float64
ID_railroad_station_walk : float64
railroad_station_avto_km : float64
railroad_station_avto_min : float64
ID_railroad_station_avto : int64
public_transport_station_km : float64
public_transport_station_min_walk : float64
water_km : float64
water_1line : object
mkad_km : float64
ttk_km : float64
sadovoe_km : float64
bulvar_ring_km : float64
kremlin_km : float64
big_road1_km : float64
ID_big_road1 : int64
big_road1_1line : object
big_road2_km : float64
ID_big_road2 : int64
railroad_km : float64
railroad_1line : object
zd_vokzaly_avto_km : float64
ID_railroad_terminal : int64
bus_terminal_avto_km : float64
ID_bus_terminal : int64
oil_chemistry_km : float64
nuclear_reactor_km : float64
radiation_km : float64
power_transmission_line_km : float64
thermal_power_plant_km : float64
ts_km : float64
big_market_km : float64
market_shop_km : float64
fitness_km : float64
swim_pool_km : float64
ice_rink_km : float64

stadium_km : float64
basketball_km : float64
hospice_morgue_km : float64
detention_facility_km : float64
public_healthcare_km : float64
university_km : float64
workplaces_km : float64
shopping_centers_km : float64
office_km : float64
additional_education_km : float64
preschool_km : float64
big_church_km : float64
church_synagogue_km : float64
mosque_km : float64
theater_km : float64
museum_km : float64
exhibition_km : float64
catering_km : float64
ecology : object
green_part_500 : float64
prom_part_500 : float64
office_count_500 : int64
office_sqm_500 : int64
trc_count_500 : int64
trc_sqm_500 : int64
cafe_count_500 : int64
cafe_sum_500_min_price_avg : float64
cafe_sum_500_max_price_avg : float64
cafe_avg_price_500 : float64
cafe_count_500_na_price : int64
cafe_count_500_price_500 : int64
cafe_count_500_price_1000 : int64
cafe_count_500_price_1500 : int64
cafe_count_500_price_2500 : int64
cafe_count_500_price_4000 : int64
cafe_count_500_price_high : int64
big_church_count_500 : int64
church_count_500 : int64
mosque_count_500 : int64
leisure_count_500 : int64
sport_count_500 : int64
market_count_500 : int64
green_part_1000 : float64
prom_part_1000 : float64
office_count_1000 : int64
office_sqm_1000 : int64
trc_count_1000 : int64
trc_sqm_1000 : int64
cafe_count_1000 : int64
cafe_sum_1000_min_price_avg : float64
cafe_sum_1000_max_price_avg : float64
cafe_avg_price_1000 : float64
cafe_count_1000_na_price : int64
cafe_count_1000_price_500 : int64
cafe_count_1000_price_1000 : int64
cafe_count_1000_price_1500 : int64
cafe_count_1000_price_2500 : int64
cafe_count_1000_price_4000 : int64
cafe_count_1000_price_high : int64
big_church_count_1000 : int64
church_count_1000 : int64
mosque_count_1000 : int64

leisure_count_1000 : int64
sport_count_1000 : int64
market_count_1000 : int64
green_part_1500 : float64
prom_part_1500 : float64
office_count_1500 : int64
office_sqm_1500 : int64
trc_count_1500 : int64
trc_sqm_1500 : int64
cafe_count_1500 : int64
cafe_sum_1500_min_price_avg : float64
cafe_sum_1500_max_price_avg : float64
cafe_avg_price_1500 : float64
cafe_count_1500_na_price : int64
cafe_count_1500_price_500 : int64
cafe_count_1500_price_1000 : int64
cafe_count_1500_price_1500 : int64
cafe_count_1500_price_2500 : int64
cafe_count_1500_price_4000 : int64
cafe_count_1500_price_high : int64
big_church_count_1500 : int64
church_count_1500 : int64
mosque_count_1500 : int64
leisure_count_1500 : int64
sport_count_1500 : int64
market_count_1500 : int64
green_part_2000 : float64
prom_part_2000 : float64
office_count_2000 : int64
office_sqm_2000 : int64
trc_count_2000 : int64
trc_sqm_2000 : int64
cafe_count_2000 : int64
cafe_sum_2000_min_price_avg : float64
cafe_sum_2000_max_price_avg : float64
cafe_avg_price_2000 : float64
cafe_count_2000_na_price : int64
cafe_count_2000_price_500 : int64
cafe_count_2000_price_1000 : int64
cafe_count_2000_price_1500 : int64
cafe_count_2000_price_2500 : int64
cafe_count_2000_price_4000 : int64
cafe_count_2000_price_high : int64
big_church_count_2000 : int64
church_count_2000 : int64
mosque_count_2000 : int64
leisure_count_2000 : int64
sport_count_2000 : int64
market_count_2000 : int64
green_part_3000 : float64
prom_part_3000 : float64
office_count_3000 : int64
office_sqm_3000 : int64
trc_count_3000 : int64
trc_sqm_3000 : int64
cafe_count_3000 : int64
cafe_sum_3000_min_price_avg : float64
cafe_sum_3000_max_price_avg : float64
cafe_avg_price_3000 : float64
cafe_count_3000_na_price : int64
cafe_count_3000_price_500 : int64
cafe_count_3000_price_1000 : int64

cafe_count_3000_price_1500 : int64
cafe_count_3000_price_2500 : int64
cafe_count_3000_price_4000 : int64
cafe_count_3000_price_high : int64
big_church_count_3000 : int64
church_count_3000 : int64
mosque_count_3000 : int64
leisure_count_3000 : int64
sport_count_3000 : int64
market_count_3000 : int64
green_part_5000 : float64
prom_part_5000 : float64
office_count_5000 : int64
office_sqm_5000 : int64
trc_count_5000 : int64
trc_sqm_5000 : int64
cafe_count_5000 : int64
cafe_sum_5000_min_price_avg : float64
cafe_sum_5000_max_price_avg : float64
cafe_avg_price_5000 : float64
cafe_count_5000_na_price : int64
cafe_count_5000_price_500 : int64
cafe_count_5000_price_1000 : int64
cafe_count_5000_price_1500 : int64
cafe_count_5000_price_2500 : int64
cafe_count_5000_price_4000 : int64
cafe_count_5000_price_high : int64
big_church_count_5000 : int64
church_count_5000 : int64
mosque_count_5000 : int64
leisure_count_5000 : int64
sport_count_5000 : int64
market_count_5000 : int64
price_doc : int64