Machine Learning Engineer Nanodegree

Capstone Project

P6: Sberbank Russian Housing Market

Report

I. Definition

Project Overview

Regression analysis is a form of math predictive modeling which investigates the relationship between variables. It answers the questions: Which factors matter most? Which can we ignore? How do those factors interact with each other? And, perhaps most importantly, how certain are we about these factors and their predictions?

The main factor that we're trying to understand or predict is a target (a dependent variable). The features (independent variables) are the factors we suppose to have an impact on the dependent variable. Using this set of variables, we generate a function that maps inputs to outputs. The training process continues until the model achieves the desired level of accuracy.

The project investigates **supervised learning** as a part of regression analysis that uses a known (training) dataset to make predictions. This dataset includes input data and response values. The supervised learning algorithms seek to build models which make predictions of the response values for a new dataset. A test dataset is used to validate the model.

Housing costs are a sphere in the real economy for applying supervised learning. They demand a significant investment from both consumers and developers. And when it comes to planning a budget—whether personal or corporate—the last thing anyone needs is uncertainty about one of their budgets expenses. Sberbank, Russia's oldest and largest bank, helps their customers by making predictions about reality prices so renters, developers, and lenders are more confident when they sign a lease or purchase a building.

Although the housing market is relatively stable in Russia, the country's volatile economy makes forecasting prices as a function of apartment characteristics a unique challenge. Complex interactions between housing features such as a number of bedrooms and location are enough to make pricing predictions complicated. Adding an unstable economy to the mix means Sberbank and their customers need more than simple regression models in their arsenal.

The project was built on the basis of the competition offered on the site https://www.kaggle.com.

Problem Statement

Sberbank is challenging programmers to develop algorithms which use a broad spectrum of features to predict real prices. Algorithm applications rely on a rich dataset that includes housing data and macroeconomic patterns. An accurate forecasting model will allow Sberbank to provide more certainty to their customers in an uncertain economy.

My choice of the solution in this situation is to select the most correlated indicators with the target variable and apply ensemble algorithms that have repeatedly shown successful results in the study of price trends in

real estate. Boosting and bagging methods combine several models at once in order to improve the prediction accuracy on learning problems with a numerical target variable.

Then I am going to explore the different types of neural networks in the sphere of regression predictions and try to achieve the same with ensemble methods level of model perfomance.

The most valuable side of this project is the investigation of real data and the attempt to approximate the predictions on them to the threshold of 0.7-0.8 for the coefficient of determination.

Metrics

The wide spectrum of popular metrics for regression was chosen and documented.

1. Explained variance regression score.

If $\hat{y}$ is the estimated target output, y the corresponding (correct) target output, and `Var` is variance, the square of the standard deviation, then the explained variance is estimated as follow:

$$\texttt{explained\_variance}(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

2. Coefficient of determination.

If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the score `R²` estimated over n_{samples} is defined as

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \bar{y})^2} \quad \text{where} \quad \bar{y} = \frac{1}{n_{\text{samples}}}\sum_{i=0}^{n_{\text{samples}}-1} y_i .$$

3. Mean squared error.

If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean squared error (MSE) estimated over n_{samples} is defined as $\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}}\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \hat{y}_i)^2.$

4. Mean absolute error.

If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean absolute error (MAE) estimated over n_{samples} is defined as $\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}}\sum_{i=0}^{n_{\text{samples}}-1}|y_i - \hat{y}_i|.$

5. Median absolute error.

If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the median absolute error (MedAE) estimated over n_{samples} is defined as $\text{MedAE}(y, \hat{y}) = \text{median}(|\ y_1 - \hat{y}_1\ |, \ldots, |\ y_n - \hat{y}_n\ |).$

Evaluation metrics capture different properties of the prediction performance: how well the model explains the target variance and makes predictions, how far the predictions are from the real values. It allows us to choose the best algorithm by comparing many indicators.

II. Analysis

Data Exploration

The datas the for the investigation is a large number of economic indicators for pricing and prices themselves (train.csv and test.csv). Macroeconomic variables are collected in a separate file for transaction dates (macro.csv). In addition, the detailed description of variables is provided (data_dictionary.txt).

```
Sberbank Russian Housing, Dataset Descriptive Statistics:

Number of houses =  30471
Number of features =  44
Minimum house price =  100000
Maximum house price =  111111112
Mean house price =  7123035.28
Median house price =  6274411.00
Standard deviation of house prices = 4780032.89
```

For practical reasons, I have not analyzed all the data and have chosen the following independent variables:

1. the dollar rate, which traditionally affects the Russian real estate market;
2. the distance in km from the Kremlin (the closer to the center of the city, the more expensive);
3. indicators characterizing the availability of urban infrastructure nearby (schools, medical and sports centers, supermarkets, etc.) ;
4. indicators of a particular living space (number of rooms, floor, etc.);
5. proximity to transport nodes (for example, to the metro);
6. indicators of population density and employment in the region of housing accommodation.

All these economic indicators have a strong influence on price formation and can be used as a basic set for regression analysis. Examples of numerical variables: the distance to the metro, the distance to the school, the dollar rate at the transaction moment, the area of the living space. Examples of categorical variables: neighborhoods, the nearest metro station, the number of rooms.

Here data outliers are, in most cases, expensive price categories. They have a strong influence on the market prices in general, so I did not exclude them from the analysis but applied the necessary method of scaling the variables RobustScaler().

We should also note that the features are not normally distributed. But the lognormal distribution looks very similar to their properties.

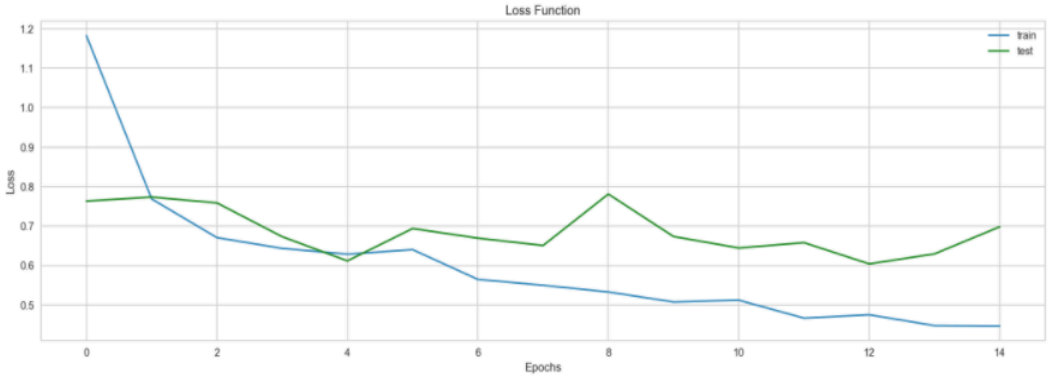
The goal of the project is to predict the price of housing using the chosen set of numerical and categorical variables. The predicted target is not discrete, for the training set all the values of this dependent variable are given, and therefore it is necessary to apply the regression algorithms of supervised learning.

The data preprocessing confirmed the assumption: these variables are in a sufficiently strong relationship with the target variable. They are used as the basis for building different types of models in several forms: only numerical variables, numeric and categorical variables transformed into numeric or binary code.

Exploratory Visualization

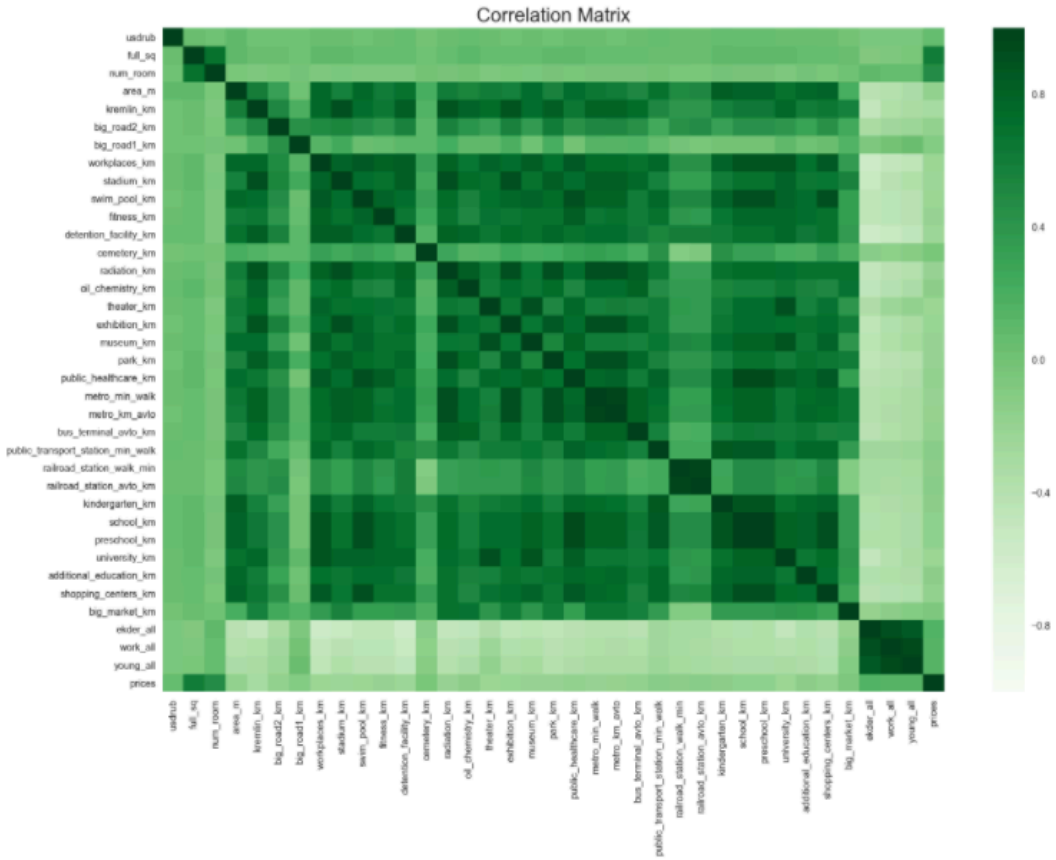
To realize the project it was necessary to use a lot of visualization tools at all stages: data tables, distributions of quantities, correlation maps, the graphical comparison of predictions and real values, representation of the feature importance for specific algorithms, operation processes and architecture of neural networks, etc.

For example: the loss function (pic.1) displays the effectiveness of neural network training, the correlation matrix (pic.2) shows the relationship between many variables, feature importance (pic.3) explains the influence of each variable on the concrete regression model.



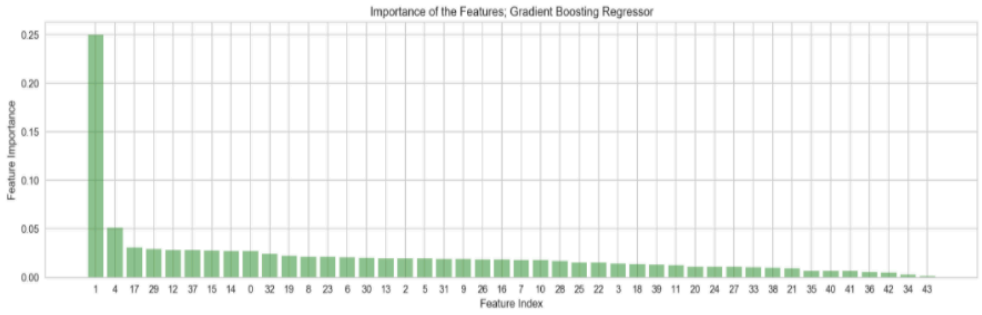
Pic.1 Loss Function

On the picture, we see two curves of decreasing training and testing loss functions. Decreasing is uneven, irregular. The graph shows that the model can be improved by smoothing out these jumps. For this goal, we can add some layers (fully-connected or others).



Pic. 2 Correlation Matrix

The correlation matrix shows very strong feature dependences one from another (darkest colors - the positive correlation, lightest colors - the negative). It means serious difficulties in improving the predictions based on this database. However, for such data sets, we have the ability to reduce the dimensionality.



Pic. 3 Feature Importance

On the picture, we can see the importance of variables and very large differences in their effect on the specific type of regressors. Thus combining regressors with considering of this influence is an additional possibility for improving models.

Algorithms and Techniques

To compare the prediction quality, I chose this set of tools.

1. ScikitLearn ensemble and neural network algorithms: Gradient Boosting Regressor, Bagging Regressor, MLP Regressor.
2. Keras: Dense, Flatten, Dropout, Convolutional, Max-Polling, LSTM layers and additional tools.
3. Numpy, Pandas, ScikitLearn: tools for data preprocessing.
4. Matplotlib, Seaborn: plots for data visualization.

In addition, I was wondering what the highest perfomance rate will be achieved by each of the presented algorithms and whether the predicted trends of price change for all used types of techniques will coincide.

The first group of algorithms was chosen from **ensemble methods**. It combines the predictions of several base estimators built with a given learning algorithm (Decision Tree) in order to improve generalizability and robustness over a single estimator. They work very well with financial data because of these characteristics.

The **Bagging Regressor** is an **averaging** ensemble method. It builds several estimators independently on random subsets of the original training set and then averages their predictions. As a result, the combined estimator is usually better than any single one because its variance is reduced.

The **Gradient Boosting Regressor** is a **boosting** ensemble method. It combines base estimators sequentially and one tries to reduce the bias of the final estimator (a powerful ensemble). The mechanism of the model consists of three important components: the loss function for checking how well our model predicts the outputs based on input values, the Decision Tree algorithms for making predictions, the additive mechanism for algorithms for minimizing the loss function. At each particular Gradient Boosting iteration, a new algorithm is trained with respect to the error that was learned so far. This procedure has the following steps: 1) add one algorithm that can reduce the loss function based on the current estimates (existing algorithms in the model are not changed); 2) use an effective procedure called gradient descent to minimize the loss; 3) repeat till the fixed number of algorithms are added or the loss reaches an acceptable level or the loss no longer improves on an external validation dataset. The result of the model training should be that predictions slowly converge toward observed values.

Neural networks such as multi-layer perceptrons (**MLP**), convolutional neural networks (**CNN**), recurrent neural networks (**RNN**) are built from layers:

- **Dense** (fully connected) layers compute the output scores, resulting in volume of size. Each neuron in these layers are connected to all the numbers in the previous volume.
- **Activation** applies the certain activation function to an output.
- **Dropout** layers consist in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.
- **Flatten** layers flatten the input and collapses it into the one-dimensional feature vector.
- **Convolutional** layers Conv1D (temporal convolution) convolve the filter with the signal, i.e. "is sliding over the signal vector, computing dot products". Here the filter is an integer, the dimensionality of the output space (i.e. the number output of filters in the convolution) and the kernel size is an integer, specifying the length of the 1D convolution window.
- **Max-Polling** layers MaxPolling1D layers perform a downsampling operation along the temporal data. Max-pooling partitions the input signal into a set of non-overlapping samples and, for each such subsample, outputs the maximum value.
- **Recurrent** Layers LSTM (Long-Short Term Memory) are a type of artificial neural network designed to recognize patterns in sequences of data, such as numerical times series. Recurrent Layers possess a certain type of memory. For example, LSTMs contain information outside the normal flow of the recurrent network in a gated cell. Information can be stored in, written to, or read from a cell, much like data in a computer's memory. The cell makes decisions about what to store, and when to allow reads, writes and erasures, via gates that open and close. Unlike the digital storage on computers, however, these gates are analog, implemented with element-wise multiplication by sigmoids, which are all in the range of 0-1. Analog has the advantage over digital of being differentiable, and therefore suitable for backpropagation.

Benchmark

The benchmark regressor among investigated models is the Gradient Boosting algorithm, it has the best level of all the evaluation metrics. We should notice that the Bagging algorithm results are really close to Gradient Boosting.

The regressor final parameters:

```
GradientBoostingRegressor(max_depth=4, n_estimators=360)
```

```
{'alpha': 0.9,
 'criterion': 'friedman_mse',
 'init': None,
 'learning_rate': 0.1,
 'loss': 'ls',
 'max_depth': 4,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 360,
 'presort': 'auto',
 'random_state': None,
 'subsample': 1.0,
 'verbose': 0,
 'warm_start': False}
```

The Convolutional Neural Networks (CNN model) for 36 numeric features demonstrates the best predictions among neural networks.

Here we can see the final CNN architecture and training process:

```
# Create the sequential model
def cnn_model():
    model = Sequential()

    model.add(Conv1D(36, 3, padding='valid', activation='relu', input_shape=(36, 1)))
    model.add(MaxPooling1D(pool_size=2))

    model.add(Flatten())
```

```
    model.add(Dense(512, activation='relu', kernel_initializer='normal',))
    model.add(Dropout(0.5))

    model.add(Dense(1, kernel_initializer='normal'))

    model.compile(loss='mse', optimizer='rmsprop', metrics=['mae'])
    return model

cnn_model = cnn_model()
# Create the checkpointer for saving the best results
cnn_checkpointer = ModelCheckpoint(filepath='weights.best.cnn.hdf5',
                                   verbose=2, save_best_only=True)
# Fit the model
cnn_history = cnn_model.fit(X_train.reshape(-1, 36, 1), y_train,
                            epochs=30, batch_size=128, verbose=0, callbacks=[cnn_checkpointer],
                            validation_data=(X_test.reshape(-1, 36, 1), y_test))
```

III. Methodology

Data Preprocessing

Data processing consisted of the following important steps:

1. deleting rows with a lot of missing data;
2. filling a small amount of missing data by linear interpolation;
3. the addition of a macroeconomic indicator;
4. transforming categorical variables into discrete numerical and binary encoded features;
5. checking the coding and eliminating the differences between the category variables in the training and test sets.

Implementation

In this project, I have used the following versions of software libraries: Scikit-Learn - 0.19.0, Keras - 2.0.4, Numpy - 1.13.1, Pandas – 0.20.2, Matplotlib – 2.0.2, Seaborn – 0.7.1.

Two ensemble Scikit-Learn algorithms (Gradient Boosting and Bagging), Scikit-Learn Multi-Layer Perceptron Regressor, three types of Neural Networks (Keras) were applied to three sets of the features (numeric, numeric and categorical, numeric and encoded categorical).

For fitting and scoring regressors I have built the simple functions. For example:

```
def regression(regressor, x_train, x_test, y_train):
    reg = regressor
    reg.fit(x_train, y_train)

    y_train_reg = reg.predict(x_train)
    y_test_reg = reg.predict(x_test)

    return y_train_reg, y_test_reg
```

The architecture of each network is implemented as a function consisting of a sequence of layers and compilation. Then the model is fitting with saving the best parameters and making predictions. For example:

```
def mlp_model():
    model = Sequential()

    model.add(Dense(1024, activation='relu', input_dim=36))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(1, kernel_initializer='normal'))

    model.compile(loss='mse', optimizer='nadam', metrics=['mae'])

    return model
```

Such a wide range of algorithms allowed to determine the approximate level of the achievable coefficient of determination of test predictions for this dataset: 69-72%. Identifying the most effective algorithms in the sphere of real financial indicators is also an important task for machine learning in general.

A detailed technical report on the algorithm parameters and the architecture of each particular model is presented in the Jupyter notebook format.

Refinement

The table with the tuning parameters for the test dataset:

Algorithms	Initial Parameters	Final Parameters	R2		MSE	
			Initial	Final	Initial	Final
Numeric Features						
Gradient Boosting Regressor	default	max_depth= 4, n_estimators=360	**0.7082**	**0.7207**	**0.5837**	**0.5587**
Bagging Regressor	default	n_estimators= 360	**0.7071**	**0.7203**	**0.5859**	**0.5594**
MLP Regressor	default	hidden_layer_sizes=(360,), max_iter=300, solver='adam', alpha=0.01	**0.6808**	**0.6969**	**0.6386**	**0.6063**
MLP	Dense(32), optimizer='rmsprop'	Dense(1024)=>Dense(128), optimizer='nadam'	**0.6536**	**0.6984**	**0.6930**	**0.6034**
CNN	Dense(64)	Dense(512)=>Dropout(0.5)	**0.6985**	**0.7113**	**0.6030**	**0.5775**
RNN	LSTM(36)	3*LSTM(144)	**0.6808**	**0.6938**	**0.6385**	**0.6125**
Numeric and Categotical Features						
Gradient Boosting Regressor	default	max_depth=3, n_estimators=396	**0.7079**	**0.7152**	**0.5844**	**0.5697**
Bagging Regressor	default	n_estimators= 308	**0.6863**	**0.7205**	**0.6275**	**0.5590**
MLP Regressor	default	hidden_layer_sizes=(396,), max_iter=300, solver='adam', alpha=0.01	**0.6790**	**0.6938**	**0.6421**	**0.6126**
MLP	Dense(64), optimizer='rmsprop'	Dense(1024)=>Dense(64), optimizer='nadam'	**0.6718**	**0.6987**	**0.6565**	**0.6028**
CNN	Dense(64)	Dense(256)=>Dropout(0.5)	**0.6889**	**0.7058**	**0.6223**	**0.5885**
RNN	LSTM(44)	LSTM(156), LSTM(624)	**0.6862**	**0.6941**	**0.6277**	**0.6120**
Numeric and Encoded Categorical Features						
Gradient Boosting Regressor	default	max_depth=4, n_estimators=318	**0.7047**	**0.7080**	**0.5907**	**0.5841**
Bagging Regressor	default	n_estimators= 159	**0.6897**	**0.7177**	**0.6208**	**0.5647**
MLP Regressor	default	hidden_layer_sizes=(318,), max_iter=150, solver='lbfgs', alpha=0.01	**0.6327**	**0.6956**	**0.7348**	**0.6088**
MLP	Dense(1024)	2*Dense(159)=>Dropout(0.1) =>2*Dense(318)=>Dropout(0.1) =>2*Dense(636)	**0.6686**	**0.6913**	**0.6630**	**0.6174**
CNN	Dense(128)	Dense(512)=>Dropout(0.5)	**0.6894**	**0.7033**	**0.6212**	**0.5935**
RNN	LSTM(636)	LSTM(159), LSTM(636)	**0.6529**	**0.7093**	**0.6943**	**0.5816**

Improvements in performance were achieved by optimizing the parameters of the algorithms or developing the structure of the neural networks. As a result, in many models, the indicator "coefficient of determination" has changed from the beginning level 0.63-0.70 to the final level 0.69-0.72 on the test data.

Hyperparameter tuning for Gradient Boosting and Bagging Regressors are pretty simple by applying the GridSearchCV() function.

For neural networks, I experimented hundreds of times and found the combination of layers and there parameters which produce about the same results with ensemble algoritms.

Here it is the most successfull model architectura for this dataset among all experiments:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_7 (Conv1D)            (None, 34, 36)            144
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 17, 36)            0
_________________________________________________________________
flatten_7 (Flatten)          (None, 612)               0
_________________________________________________________________
dense_162 (Dense)            (None, 512)               313856
_________________________________________________________________
dropout_20 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_163 (Dense)            (None, 1)                 513
=================================================================
Total params: 314,513
Trainable params: 314,513

Non-trainable params: 0
```

IV. Results

Model Evaluation and Validation

All the measurements listed in the section "Metrics" were used to evaluate the perfomance of models.

The best indicators for ensemble algorithms.

```
<_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_>
Numeric Features; Gradient Boosting Regressor
<_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_>
EV score. Train:  0.86189746402
EV score. Test:  0.720854054972
----------
R2 score. Train:  0.86189746402
R2 score. Test:  0.720764275765
----------
MSE score. Train:  0.251150449123
MSE score. Test:  0.558723845618
----------
MAE score. Train:  0.31458911313
MAE score. Test:  0.400128339345
----------
MdAE score. Train:  0.174402117839
MdAE score. Test:  0.199730980341
```

The best indicators for neural networks.

```
<_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_>
Numeric Features; CNN Model
<_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_>
EV score. Train:  0.735439071809
EV score. Test:  0.711557145756
----------
R2 score. Train:  0.735350946162
R2 score. Test:  0.711347161402
----------
MSE score. Train:  0.481285359893
MSE score. Test:  0.577566586338
----------
MAE score. Train:  0.400953623408
MAE score. Test:  0.427550065202
----------
MdAE score. Train:  0.214423935148
MdAE score. Test:  0.22883767204
```

In the analysis, I did not exclude outliers in general, so the special sensitivity of models to the addition of other data (including noises or outliers) should not be expected. It can be confirmed by applying the algorithms for the whole dataset. As it can be seen this experiment has very similar results with the smaller training set.

```
<_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_>
 Gradient Boosting Regressor
<_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_><_>
EV score: 0.851729559483
----------
R2 score: 0.851729559483
----------
MSE score: 0.273663122104
----------
MAE score: 0.324355312761
----------
MdAE score: 0.17539487972
```

The correlation coefficient between predictions and real values within 0.69-0.72 in the test case is high enough for a real data set (more than 0.6). During work over the project, I had the impression that the coefficient cannot be improved above 0.75 for this particular data set using any algorithms.

Justification

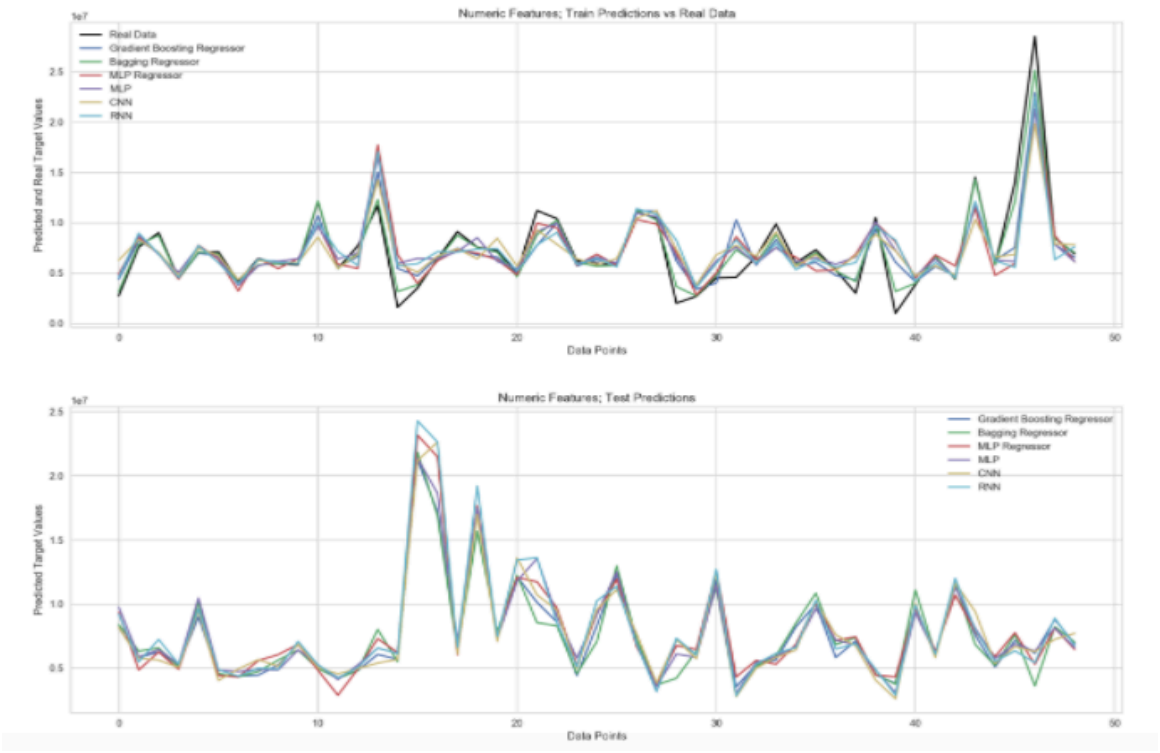
Participation of the project in the competition allowed to improve the constructed models, compare indicators and algorithms with other competitors, discuss the process and evaluate the results. The competition metric was the RMSLE (Root Mean Squared Logarithmic Error). I did not use in my work but my models demonstrated enough well results by this evaluation method also. It confirmed the effectiveness of algorithms again.

The winner had the RMSLE = **0.30087**, I had the RMSLE = **0.32766**.

V. Conclusion

Free-Form Visualization

As a final visualization, I chose the image of the predictions of all models on a single graph. As it can be seen from the illustrations, the predictions are very close to each other and determine the overall price dynamics quite clearly. This is an additional confirmation of the reliability of predictions: in case of erroneous conclusions, all models would hardly have demonstrated the same trend.



Pic. 4. Display all predictions

Reflection

The prediction of financial values is quite complex due to the strong dependence of the indicators on each other, the influence of the time factor and uncertainty. To achieve a greater approximation to real data is one of the closest and achievable tasks of machine learning.

The project database is similar to the well-known and well-studied the Boston Housing Dataset. Therefore, it was easy to start implementing the project by applying similar methods. In the course of working on the project, I experienced much more regression algorithms and neural networks than presented in the program part, then just shortened the list, leaving the most effective. Working on the data also did not present any particular difficulties: firstly I cleaned and reduced the base, then tried to use only the numerical variables, and finally added categorical ones and compared the results.

The most interesting aspect for me was to work with the project precisely because of the large range of variables in real data and the possibility to advance the understanding this field of activity.

In general, the project implementation process included the following steps: data processing and choice of variables, application of ensemble methods, improvement of their parameters, construction of neural networks and their development, model evaluation by metrics and a unified representation of all predictions.

Improvement

There are many possible ways to improve the modeling: studying of other sets of variables (maybe some variables with important information were lost), combining of existing algorithms in ensembles (to catch the trend more effective), developing the architecture of built neural networks in the project (improving structures allow to analyze more deeply), applying the existing neural networks with a complex structure from the external sources (for the same reason), etc.

VI. Bibliography

1. Amy Gallo. A Refresher on Regression Analysis. Harvard Business Review, 2015.
2. Model evaluation: quantifying the quality of predictions (http://scikit-learn.org/stable/modules/model_evaluation.html)
3. Keras: The Python Deep Learning library (https://keras.io/).