House Prices: Advanced Regression Techniques

Kaszás Bálint

January 22, 2019

1 Introduction

The goal of this project is to predict the selling price of houses. The dataset that is used for training [1] contains data for residential homes sold in Ames, Iowa. The point is to go beyond the standard characteristics like the net area of the property and number of bedrooms.

First, I will perform preliminary data exploration, and will try to isolate the most important features by computing their correlation with the sale price. I will also project the dataset to a smaller dimension using PCA and TSNE.

Then, I use several regression techniques for the prediction of the sale price. Starting from the simple k nearest neighbor estimation, moving on to linear regression with subset selection and shrinkage. I also train a neural network, and finally use a tree based model with gradient boosting. The performance of these models is also assessed. The code used in the project can be found on [2].

2 Data exploration

The dataset contains 79 feature variables, the full list can be seen on [1]. Naturally, there are quite a few missing values in the dataset. From a preliminary look, the variables listed below contain a considerable amount of missing data (i.e. for more than 10 % of the whole dataset, these variables are missing).

- 'LotFrontage' The linear feet of street connected to property
- 'Alley' The type of alley access
- 'FireplaceQu' The fireplace quality
- 'PoolQC' The pool quality
- 'Fence' The fence quality
- 'MiscFeature' Miscellaneous feature not covered in other categories

For simplicity, I discarded these variables for the purposes of further analysis. After this, some missing data was left, but discarding them only yielded an approximately 5 % loss in total datapoints.

The goal of the project is to predict the price, at which the given house will sell, given its condition. First, let us look at the distribution of the sale prices in the given dataset that we will use for training. Figure 1 shows the histogram constructed from the target variable of the training set. The distribution is highly skewed to the left. It is also reflected in the difference between the average and median sale prices.

- Average sale price is 186761 $
- Median sale price is 168500 $
- Standard deviation is 78884 $

The dataset contains various numeric and non numeric features. It is worth looking at the numeric variables first. I computed the correlation of each of these variables with each other. Figure **??** shows the 5 variables, that correlate best with the target variable, sale price.

We obtain the best correlation for the variables representing overall quality, the size of the living area, the capacity of the garage, the size of the garage and the 1st floor's area.

We see the highest correlation coefficient between sale price and overall quality, which means that this variable is a true representation of the house's worth. We do not see heavy correlation between the variables, with the exception of garage capacity and garage area, which is trivial.

To better visualize the most important variables, Fig. 3 shows the distribution of their distributions and their relation to sale price on a scatter plot. The positive correlation is clearly visible for all of the variables, with the possible exception of garage capacity as expressed in the number of cars. This is because the variable itself has a limited range (has values 0, 1, 2, 3 or 4).

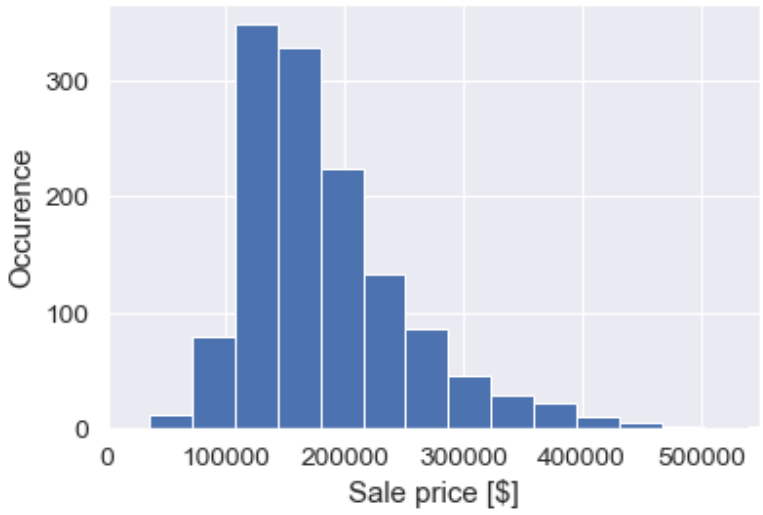

Figure 1: Histogram of the sale prices in the dataset used.

2.1 Dimension reduction

As the last step of exploratory data analysis, I performed dimension reduction using two methods, principal component analysis (PCA) and t-distributed stochastic neighbor embedding (TSNE) [3]. Figure 4 shows the result of these two methods, in the plane of the two most important dimensions.

TSNE analysis yields a much clearer picture than principal component analysis. We see a fairly good separation of datapoints. This comes from the difference in the overall quality of the houses, as the color coding shows.

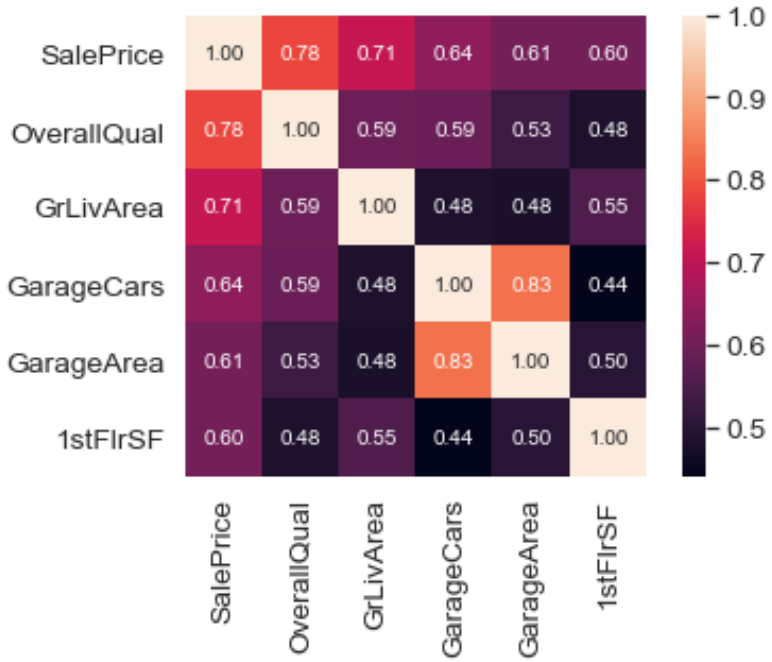

Figure 2: The upper corner of the correlation matrix, containing the 5 variables that correlate best with sale price. In each cell, the corresponding coefficient is displayed. The cells are also color coded (see colorbar).

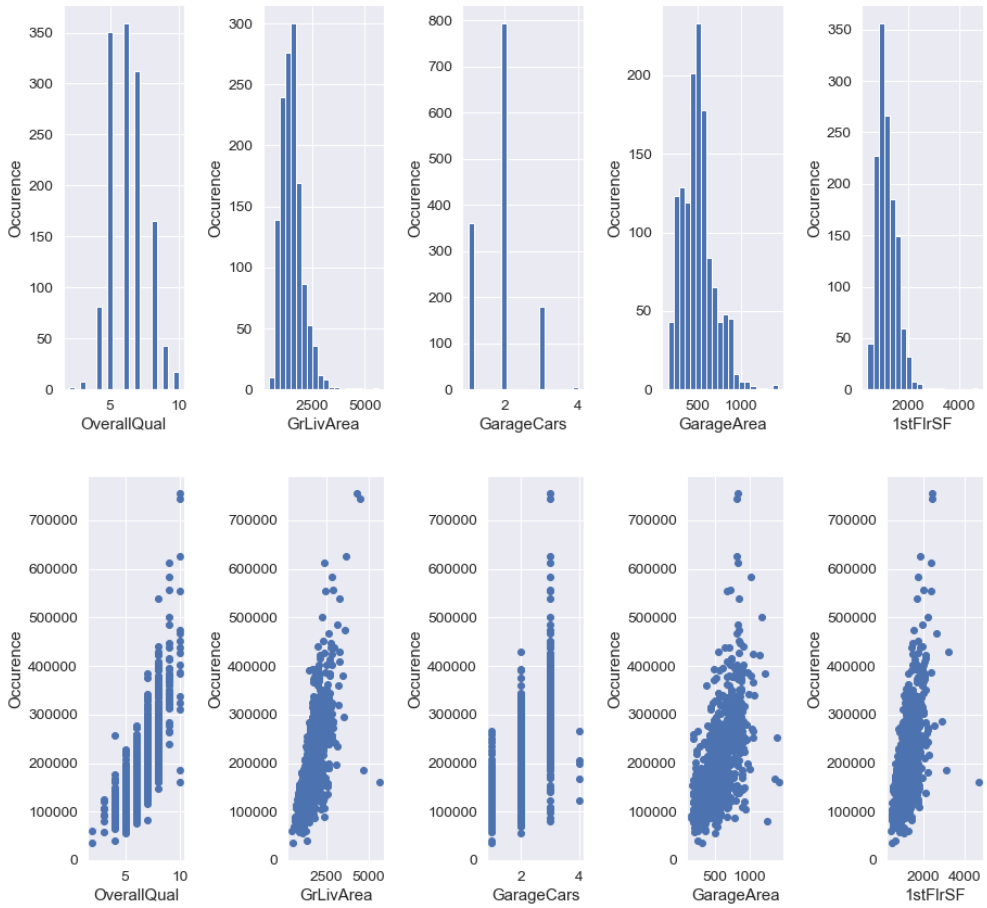

Figure 3: Top row: Distribution of the 4 most important variables. Bottom row: The sale price displayed as a function of the 4 most important variables.

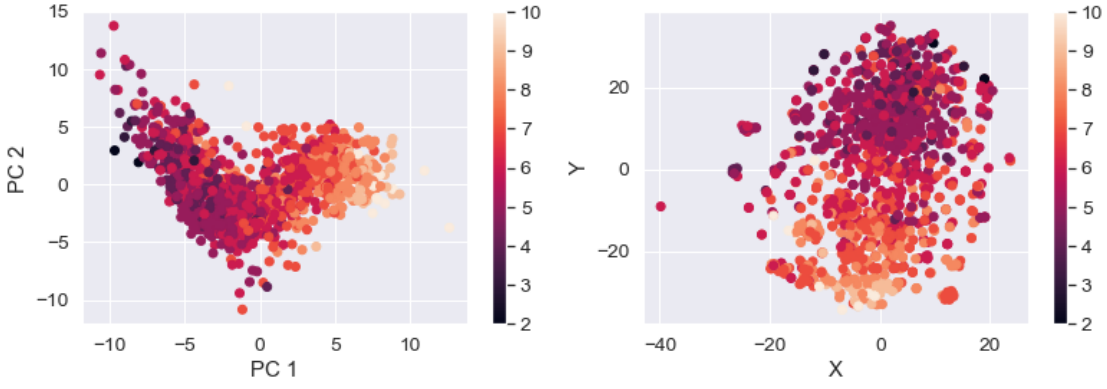

Figure 4: The result of dimension reduction. On the left the first and second principal components, and on the right, the first and second TSNE components are displayed. The colors indicate the overall quality of the house.

3 Regression techniques

To attempt to predict the price of a house using the variables available in the dataset, I will try a number of different methods.

For the regression, I encoded the non numeric data with the pandas library's

```
get_dummies()
```

command. This essentially does a one-hot encoding, and adds the encoded vectors to the existing dataframe. Furthermore, to make handling the data easier I performed standard scaling in each variable.

3.1 KNN estimation

The first model that I try to use for house price prediction is the k nearest neighbor estimation. To predict a price of a datapoint, this method simply takes the average price of the points that are near the given datapoint. The only parameter is k, the number of nearby points used for the averaging.

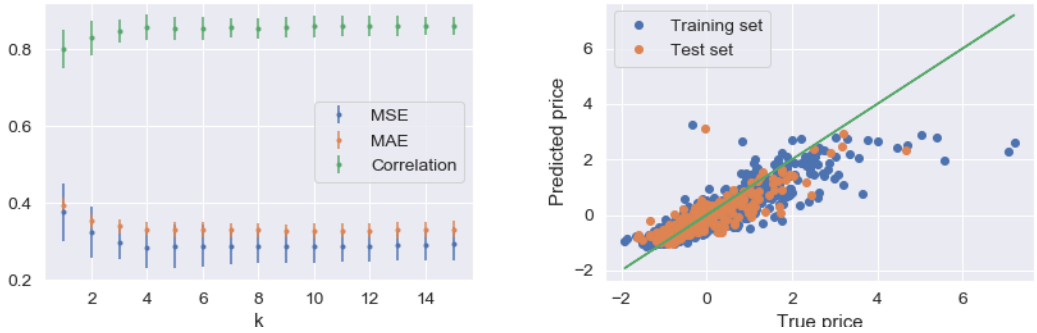

Figure 5: Left: analysis of the performance of the model as a function of parameter k. Right: Scatterplot showing the predicted price as a function of true price.

Figure 5 shows the dependence of the accuracy of the model as we increase the number of neighbors used. Here and in the following, I use 3 metrics to quantify the accuracy:

- Mean Squared Error: MSE $= \frac{1}{n}\sum_i (X_i - \bar{X}_i)^2$
- Mean Absolute Error: MAE $= \frac{1}{n}\sum_i |X_i - \bar{X}_i|$
- Pearson correlation coefficient.

To select the best KNN model, I used 5 fold cross validation. The metrics are computed in each fold and then averaged. The optimal number of neighbors turned out to be k = 13.

Using a train-test split of 4 to 1, I trained the model with k = 13. The result of the prediction is shown in the right panel of Fig. 5. We see that the model seems to undervalue houses most of the time, since the majority of the data lies below the $y = x$ line. This effect is even stronger for houses that have a high selling price.

3.2 Iterative forward selection

Next, let us turn to linear regression models. Due to the encoding of the categorical variables, the dataset now contains over 250 features. Out of these features, we want to select as few as possible, that properly predict the sale price. The fastest way to select the best features is to perform an iterative forward selection.

It begins with a model containing only one predictor, and one at a time, adds further predictors. In each step, it chooses the predictor that has the most contribution to the model. This is measured

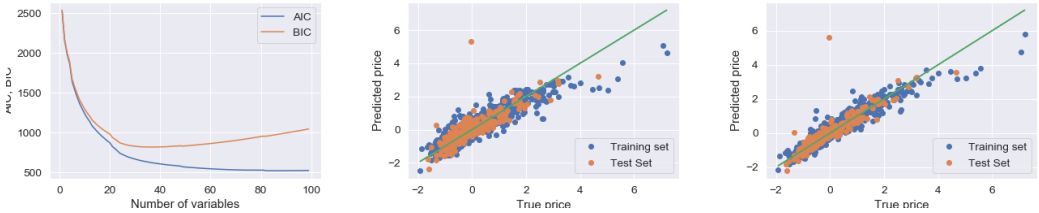

Figure 6: The result of iterative forward selection. In the left panel the AIC and BIC values are displayed, as more features are added to the model. In the middle (right) panel, the predictions of a linear model with 5 (25) features is shown.

by the adjusted R^2. The popular libraries do not contain iterative forward selection, I used the implementation of [4].

Figure 6 shows the result of iterative forward selection. In the left panel the values of AIC and BIC is computed as a function of features used in the model. These metrics attempt to quantify the information loss that is caused by using a simpler model. The minimum of Bayesian information lost (BIC) [3] is attained by a model using 25 predictors. The prediction is seen in the right panel. For comparison, I also show a much simpler model, which uses only the 5 most important features. The prediction is shown in the middle panel.

# of predictors	MSE	MAE	Correlation
5	0.24±0.07	0.31 ± 0.01	0.87 ± 0.04
25	0.18±0.09	0.22 ± 0.01	0.91 ± 0.05

Table 1: Comparison of linear models with different number of predictors.

Using a 5 fold cross validation, it is possible to estimate the accuracy of our models. Table 1 compares a linear model with 5 predictors to a linear model with 25 predictors, which is the optimal number of features. The 5 most important features, that we get from forward selection are

- 'OverallQual': Overall quality of the house
- 'GrLivArea': The living area
- 'BsmtQual_Ex': Whether the basement's height is over 100 inches
- 'RoofMatl_ClyTile': Whether the roof's material is clay /tile
- 'BsmtFinSF1': The area of the basement

3.3 Ridge and Lasso regression

Another approach to reducing the model's complexity and avoiding overfitting is to introduce shrinkage. This is implemented by the so called ridge and lasso regression. Here, additional terms are included in the cost function that is being minimized. These terms are for penalizing the magnitude of the coefficients. The difference between them is that ridge uses the L^2 norm of the coefficients, while lasso uses the L^1 norm.

This means that terms $\alpha \sum_i \beta_i^2$ or $\alpha \sum_i |\beta_i|$ are added to the loss function.

To judge the models, we investigate their accuracy as a function of the parameter α, which controls the penalty. Figure 7 shows the dependence of the mean squared error on α for both Ridge and Lasso models, using 5 fold cross validation.

Using the optimal parameters, we get nearly the same predictions as the model obtained by iterative forward selection.

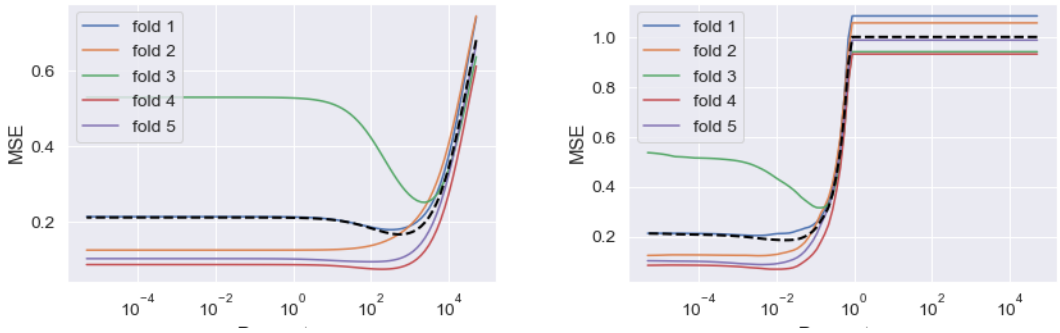

Figure 7: Dependence of accuracy on shrinkage parameter (α). In the left (right) panel the Ridge (Lasso) model is shown.

3.4 Neural network

The next model I used for prediction is a neural network. After experimenting with various simple architectures, which didn't perform well enough, I found the kernel [5]. Here, regularizers were also used to shrink the layer parameters in a similar way as in ridge or lasso regression. Using the architecture presented in the kernel, the predictions presented in Figure 8 are obtained.

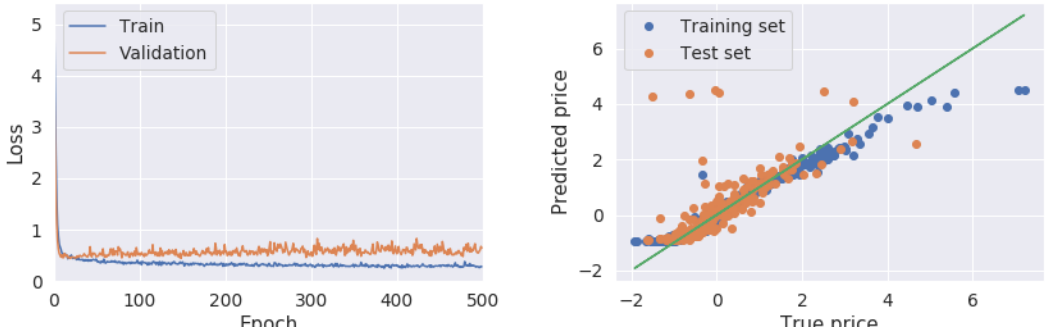

Figure 8: Predictions of house prices using a neural network. In the left panel, the model accuracy is shown during the training.

I computed the relevant metrics for the neural network too, so that it can be compared to the methods presented above. Using this particular architecture, we get worse predictions than with any of the other models used in the project.

- MSE = 0.507
- MAE = 0.311
- The correlation coefficient is 0.75

3.5 Gradient tree boosting

The final model is the gradient tree boosting method. The tree's internal nodes consist of the possible values of the input variable, and its the leaves are the values of the target variable (the sale price of the house in our case). It turns out that using gradient boosting, i. e. taking an ensemble of decision trees increases the performance dramatically [6]. The left panel of Fig. 9 shows the model's prediction.

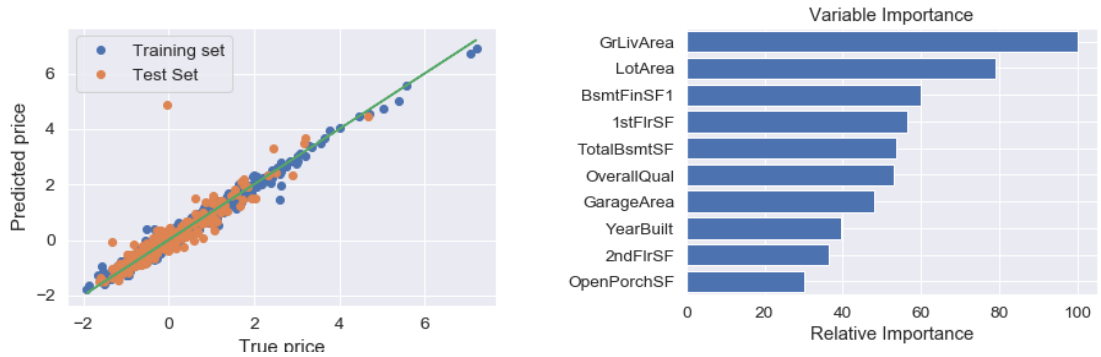

Figure 9: Prediction based on the gradient tree boosting method. In the left panel the predicted prices are shown as a function of the true prices. In the left panel the ten most important features are shown arranged by importance.

We are also able to discern the most important features, these are shown in the right panel of Fig. 9. The most important feature is the living area, the rest of the features are shown in decreasing order of importance.

The gradient boosted decision tree produces remarkable accuracy. This is also reflected in the metrics computed from five fold cross validation.

- MSE = 0.12 ± 0.03
- MAE = 0.20 ± 0.01
- The correlation coefficient is 0.94 ± 0.01

4 Conclusion

In this project, I used regression techniques to predict the prices of houses. The models were evaluated by computing the mean squared error, the mean absolute error and the Pearson correlation coefficient between the true and the predicted prices. The following table contains the summary of the results. The standard deviation of the values are computed from 5 fold cross validation.

Model	MSE	MAE	Correlation
KNN (k = 13)	0.29±0.04	0.33 ± 0.02	0.86 ± 0.02
Linear regression (n = 5)	0.24±0.07	0.31 ± 0.01	0.87 ± 0.04
Linear regression (n = 25)	0.18±0.09	0.22 ± 0.01	0.91 ± 0.05
Ridge regression	0.16± 0.06	0.22 ± 0.01	0.91 ± 0.04
Lasso regression	0.18±0.09	0.21 ± 0.01	0.91 ± 0.05
Neural network	0.51	0.31	0.75
Gradient tree boosting	0.12 ± 0.03	0.20 ± 0.03	0.94 ± 0.01

Table 2: Comparison of regression models used in the project.

The metrics presented in Table 2 show that the best results are obtained with the gradient boosted tree based method. Linear regression models on 25 predictors, or Ridge/Lasso regression show nearly the same performance, and come fairly close to the tree based regression. The neural network and k nearest neighbor estimation performed significantly worse.

4.1 Anomalies

However, all models agree that there are some houses that have a predicted price that is much lower than the sale price. These are the two most expensive houses in the dataset. They are excellent quality houses with huge living areas.

- House # 1183: With overall quality 10 and a living area of 4476. Sold for 745000 $.
- House # 632: With overall quality 10 and a living area of 4316. Sold for 755000 $.

This anomaly can be understood if we consider the highly skewed dataset, shown in Fig. 1. The more expensive houses are greatly underrepresented, and this is why the model cannot give sufficiently accurate predictions in this regime.

References

[1] `https://www.kaggle.com/c/house-prices-advanced-regression-techniques`

[2] `https://github.com/balintkaszas/houseprice`

[3] Hastie, T.; Tibshirani, R. Friedman, J. (2001), *The Elements of Statistical Learning* , Springer New York Inc. , New York, NY, USA .

[4] `https://xavierbourretsicotte.github.io/subset_selection.html`

[5] `https://www.kaggle.com/ironfrown/deep-learning-house-price-prediction-keras`

[6] `https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting`