American University of Armenia

Zaven & Sonia Akian College of Science and Engineering

CS 251/340, Machine Learning Course Project

House Price Prediction

Team: Tatevik Jalatyan, Mariam Hayrapetyan

Spring, 2021

Abstract

House price is strongly correlated to factors such as location, area, etc. and this information can be used to predict individual house price. This study includes eight machine learning algorithms to examine a data sample of about 1300 housing transactions. The comparison of the models is also included. SVR, XGBoost and LinearSVR had the best performance in terms of predictive power. The performance metrics: MSE and MAE were small because of the log transformation of the target variable but still it was possible to use them for comparisons.

Keywords: House Price Prediction, Machine Learning algorithms, SVM, RF, XGBoost

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Setting, Project Motivation and Description

The problem that is discussed in this study is predicting the house price based on the features that it has. This is a very practical problem that has been solved by different models and based on different datasets. But on the dataset that is used in this study, there are a few methods applied, and the aim of this project aims to develop a model for a house price prediction, using different methods like Linear Regression, Tree-based models, Random Forest, Ridge, Lasso, XGBoost, Support Vector Regression (SVR).

Many studies have been performed based on this problem which included many machine learning methods and also combinations of methods. In their article, Tang et al. [HTW21] used stochastic gradient descent (SGD) based support vector regression (SVM), random forest (RF), and gradient boosting machine (GBM) for the same problem. They concluded that on the given dataset, RF and GBM were able to generate comparably accurate price estimations with lower prediction errors compared with the SVM results. Zhou Yichen [Zho20], in her article, has applied different models such as linear regression, lasso regression, random forest, and XGBoost. Comparison of models showed that RMSE and R squared of Lasso was not good, RMSEof the random forest was relatively low, but it was also overfitting, the last model was XGBoost, and as RMSEand R squared seemed good, it was selected as a final model. Truong et al. [Tru+20] applied three different machine learning methods: Random Forest, XGBoost, and LightGBM, and two techniques in machine learning: Hybrid Regression and Stacked Generalization Regression for house price prediction. They declared that different methods had their pros and cons. The Hybrid Regression method was better than the three previous methods due to the generalization. The Stacked Generalization Regression method was the best choice regarding accuracy. But both of them had high time complexity.

Chapter 2

Data and Preprocessing, Performance Measurement

2.1 Dataset

The data source for this study is the House Price dataset for Ames City in the United States, available in Kaggle public source. The dataset includes 1314 entries, each with 79 features and a target variable(Price). Data consists of 43 categorical and 36 numerical features. See figure 2.1 for the distribution of house sale prices.

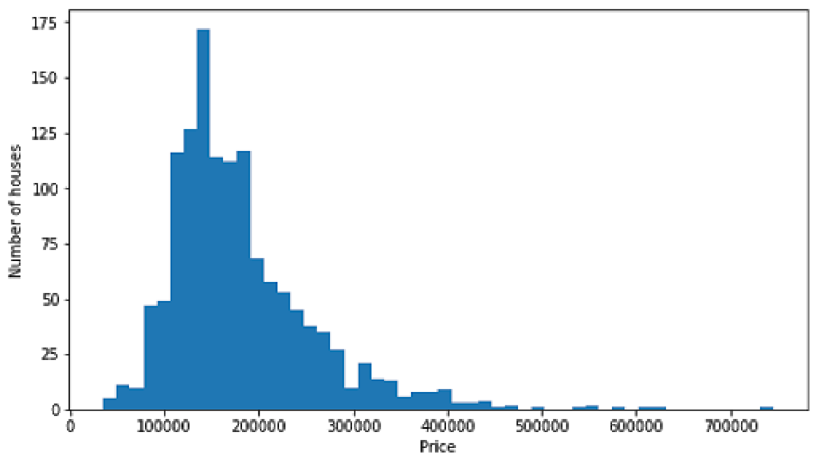


Figure 2.1: Distribution of Sale Prices

Some visualization analysis was performed for understanding the relationships between features and Sale Price (See Figure 2.2 and Figure 2.3). Correlations between Sale Price and features were calculated and the features having an absolute value of correlation coefficients higher than 0.5 were chosen for visualizations.

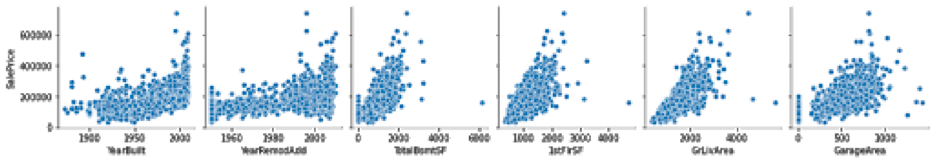


Figure 2.2: Relationship between Sale Price and highly correlated continuous features

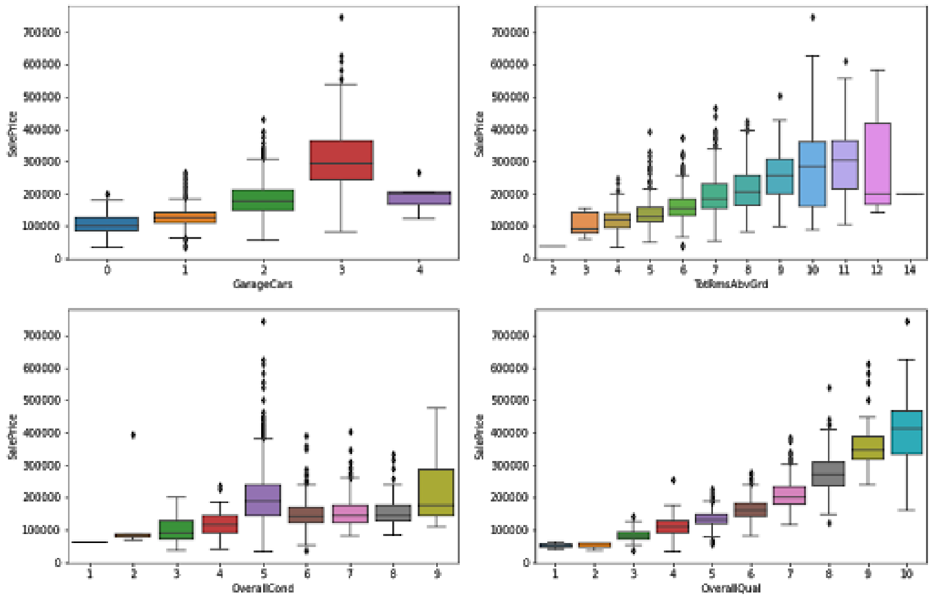


Figure 2.3: Relationship between Sale Price and highly correlated discrete features

2.2 Preprocessing and Feature selection

For the initial preprocessing, it was decided to drop columns where most of the values are missing(1), replace missing values of numerical features with the mean(2), replace missing values of categorical features with the most repeated value(3), convert the categorical features into one-hot vectors(4). As a result, the data has been transformed from 80 columns to 271. Before applying models, the data was normalized where necessary(5), and the target was converted into a log of the sale price(6). The data was split into train and test sets(7).

First, linear regression was performed on the dataset with all features, which end up with a high R-squared score and a high Mean squared error. So, feature selection was applied to select the most important features. The correlation between each regressor and the target was computed then it was converted to an F score than to a p-value. The features were selected based on these p-values using a threshold. Also, the SelectKBest function from sklearn was applied, which worked in the same way. After applying Lasso regularization, some coefficients became zero; we chose features whose coefficients were not zero and tried to use all models with these features. Some models perform better with these new set of features and some with the features based on the p-value.

2.3 Performance Measurement

The performance of each model was measured using MSE (Mean Squared Error) and MAE (Mean Absolute Error) scores, which measure the distance between the actual data and the predicted data. We chose the models that have high R^2 (near 1) and low MSE and MAE.

Chapter 3

Algorithms and Models

3.1 Algorithms and Models

The first model was the Linear Regression model with all features. After selecting some features based on p-value and another set after Lasso feature selection, different models have been applied all with sci-kit-learn library except Extreme Gradient Boosting algorithm for which XGBoost library was used.

3.1.1 Linear Regression

Given a dataset, the linear regression model assumes that the relationship between the dependent variable y and feature vector X is linear. ϵ, which is a random variable with normal distribution, adds noise to the linear relationship between y and X. The model has the form: $Y = X\beta + \epsilon$.

3.1.2 Lasso and Ridge Regressions

Lasso regression performs both variable selection and regularization in order to enhance the prediction accuracy. The objective of this method is to minimize: $w^{\hat{Lasso}} = \underset{w \in R^{(D+1)}}{\text{arg max}} \{RSS + \lambda \sum_{n=1}^{D} |w_k|\}$, where λ is a hyperparameter. Ridge regression is also a regularization method. The objective of which is to minimize: $w^{\hat{Ridge}} = \underset{w \in R^{(D+1)}}{\text{arg max}} \{RSS + \lambda \sum_{n=1}^{D} w_k^2\}$, where λ is a hyperparameter. These methods help to reduce overfitting. Since the linear regression was observed to have high variance, Lasso and Ridge regressions were a good choice to solve this issue.

3.1.3 Support Vector Regression

SVR solves the following optimization problem:
$min_{w,b,\xi,\xi^*} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i + C\sum_{i=1}^{m} \xi_i^*$, subject to

$w^T\phi(x^{(i)} + b - y^{(i)} \leq \epsilon + \xi_i,$
$y^i - w^T\phi(x^{(i)} - b \leq \epsilon + \xi_i^*,$
$\xi, \xi^* \geq 0, i = 1, ..., m$

SVR with both linear and RBF kernels have been used. So, instead of $\phi(x)$, $K(x,z) = \phi(x)\phi(z)$ can be used. For RBF kernel K(x, x) = exp($\gamma||xx'||^2$); for linear kernel K(x, x) =x^T · x.

3.1.4 K-Nearest Neighbors Regression

KNN Regression algorithm includes choosing hyperparameter k and distance metric, finding nearest k points from feature vector x and returning the average of the labels of k nearest neighbors:
$y = \frac{1}{k}\sum_{x_j \in NN_k(x)} y_j$, where k is a hyperparameter and $NN_k(x)$ is the set of k nearest points from x.

3.1.5 Decision Tree Regression

In the Decision Tree Regression Problem, the Variance is used as the Impurity Measure:
i(N) = Var(Data in N) =$\frac{1}{k}\sum_{y_i \in N}(y_i - c)^2$, where k is the number of Datapoints in the node N and c is the average of y_i for all observations in N. The split must be chosen which maximizes the Drop in Impurity:
$\Delta i(N) = i(N) - p_L \cdot i(N_L) - p_R(N_R)$, where N_L and N_R are the left and right child nodes of N, p_L and p_R are the weights of the left and right child nodes.

3.1.6 Random Forest Regression

Random Forest consists of a large number of individual decision trees that operate as an ensemble. The steps of the algorithm are taking k bootstrapped samples of the dataset, training a Decision Tree Regressor for each bootstrapped sample, when constructing each Tree randomly sampling m features and choosing the best split among them, finally averaging the obtained predictions.

3.1.7 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a decision tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. This method was chosen because XGBoost is the leading model for working with standard tabular data. [Zho20] This method repeatedly builds new models and combines them into an ensemble model. It starts by calculating the errors for each observation in the dataset then builds a new model to predict those and adds predictions from this error predicting model to the ensemble of models.

Chapter 4

Results and Conclusion

4.1 Experiments and Results

4.1.1 Linear Regression

Linear regression with all features performed very poorly with -8618817026985383 % R^2 on the test set and 94.8 % R^2 on the train set. The MSE for the test was very high, and MSE for the train was nearly 0, which is a sign of overfitting. The result of feature selection based on p-value ended up with a set of 50 features most correlated with the Sales Price variable. Linear regression was applied with both a 50-feature set and features selected by Lasso. The best model was with features chosen by Lasso with 90.9 % R^2 for the test set and 86.9% R^2 for the train test. MSE for the test was 0.0144, and MSE for train ws 0.0207.

4.1.2 Lasso Regression

For Lasso Regression first, cross validation was performed with ten folds to choose the best alpha value from the given set of values. Then the chosen alpha value was used to train Lasso Regression on the training dataset with all features. The performance of the model was good; R^2 for the test was 90.7%, MSE and MAE for the test were 0.0146 and 0.088, respectively. Also, Lasso was used for feature selection purposes. The Lasso method penalizes the coefficients of the regression variables, shrinking some of them to zero. So, all features which had non-zero regression coefficients were selected.

4.1.3 Ridge Regression

Similar to Lasso Regression, for Ridge Regression, alpha was chosen by cross validation with ten folds, and it was used to train Ridge regression on the training dataset with all features. R^2 for the test was 90.6%, MSE and MAE for the test were 0.0147 and 0.0818, respectively.

4.1.4 Support Vector Regression

For the Support Vector Regression model, Grid Search Cross Validation has been applied for choosing the best C (Hyperparameter, which controls error) and Gamma (Hyperparameter which decides that how much curvature we want in a decision boundary) based on best performance by negative mean squared error. By adding C and Gamma scores, we have obtained better results, then we decided to standardize the data, and we have added StandardScaler. The best result was SVR with 0.001 gamma, C of 10, RBF kernel, with StandardScaler in the pipeline, and with features selected by Lasso regression. The final result has 91.9% R^2 for the test, 91.3% R^2 for the train, 0.0127 MSE for the test, and 0.0137 for the train. Additionally, LinearSVR model has been applied with StandardScaler and with default parameters. The model with features selected by Lasso regression gave 91.08% R^2 for the test, 82.2% R^2 for the train, 0.0140 MSE for the test, and 0.0281 for the train.

4.1.5 K-Nearest Neighbors Regression

For K neighbors, Regressor K fold cross-validation has been applied, the data were randomly divided into k equal size subsamples, and for each K accuracy score on the test set was calculated. The K with the highest accuracy score was chosen, and the best model was with the chosen k, which was 7, on the data with 50 features based on the p-value. The results for the test are 76% R^2 and 0.0378 MSE, and for the train, R^2 is 79.7%, and MSE is 0.0321.

4.1.6 Decision Tree Regression

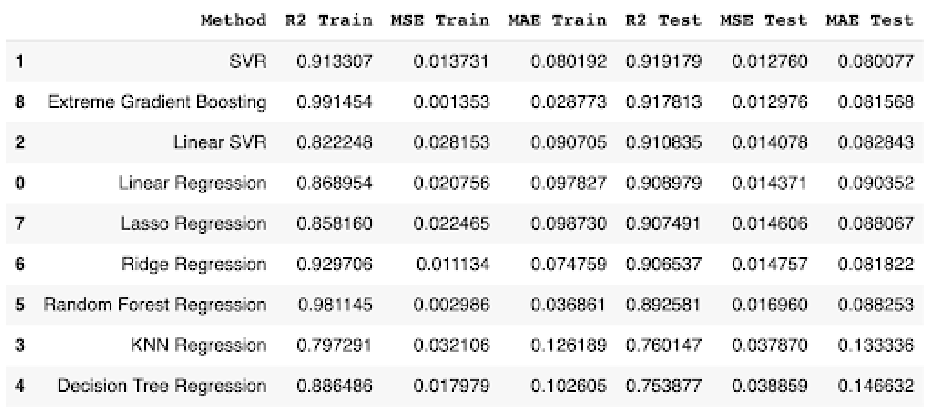
Grid Search Cross-Validation has been applied for the Decision Tree for choosing the best parameters for the maximum depth of the tree, the minimum number of samples required to be at a leaf node, the minimum weighted fraction of the sum total of weights required to be at a leaf node, the number of features to consider when looking for the best split and the maximum number of leaf nodes. The best model was chosen based on negative mean squared error and had 75.3% R^2 for the test set, 88.6% R^2 for the train, 0.0388 and 0.0179 MSE for test and train, respectively. The best results of the model were on the data with 50 features.

4.1.7 Random Forest Regression

For Random Forest Grid Search, Cross Validation has been applied, too. For this time, parameters were the maximum depth of the tree, the minimum number of samples required to be at a leaf node, the number of features to consider when looking for the best split, the minimum number of samples required to split an internal node, and the number of trees in the forest. The best model was chosen based on negative mean squared error and had 89.2% R^2 for the test set, 98.1% R^2 for the train, 0.0169, and 0.0029 MSE for the test and train, respectively. The best results of the model were on the data with 50 features.

4.1.8 Extreme Gradient Boosting

For Extreme Gradient Boosting Grid Search, Cross Validation has been applied for choosing parameters like the maximum depth of the tree, Boosting learning rate, Minimum sum of instance weight (hessian) needed in a child, subsample ratio of the training instances, the subsample ratio of columns when constructing each tree, number of gradient boosted trees equivalent to the number of boosting rounds. The best model was chosen based on negative mean squared error and had 91.7% R^2 for the test set, 99.1% R^2 for the train, 0.0129, and 0.0013 MSE for test and train, respectively. The best results of the model were on the data with 50 features.



	Method	R2 Train	MSE Train	MAE Train	R2 Test	MSE Test	MAE Test
1	SVR	0.913307	0.013731	0.080192	0.919179	0.012760	0.080077
8	Extreme Gradient Boosting	0.991454	0.001353	0.028773	0.917813	0.012976	0.081568
2	Linear SVR	0.822248	0.028153	0.090705	0.910835	0.014078	0.082843
0	Linear Regression	0.868954	0.020756	0.097827	0.908979	0.014371	0.090352
7	Lasso Regression	0.858160	0.022465	0.098730	0.907491	0.014606	0.088067
6	Ridge Regression	0.929706	0.011134	0.074759	0.906537	0.014757	0.081822
5	Random Forest Regression	0.981145	0.002986	0.036861	0.892581	0.016960	0.088253
3	KNN Regression	0.797291	0.032106	0.126189	0.760147	0.037870	0.133336
4	Decision Tree Regression	0.886486	0.017979	0.102605	0.753877	0.038859	0.146632

Figure 4.1: Performance of the Models

4.2 Conclusion

Various price prediction models are being used for predicting the price. In this work, different machine learning algorithms were applied, and comparisons between the models were done comparing R^2 in the first place and additionally by mean absolute and mean squared errors. The best model, which is the Support Vector Machines(SVR) model, has an accuracy of 91.9%. The performance of all algorithms you can see in Table 1. In Figure 4, it can be seen the difference between log transformations of actual prices and predicted ones for the best performed four algorithms.

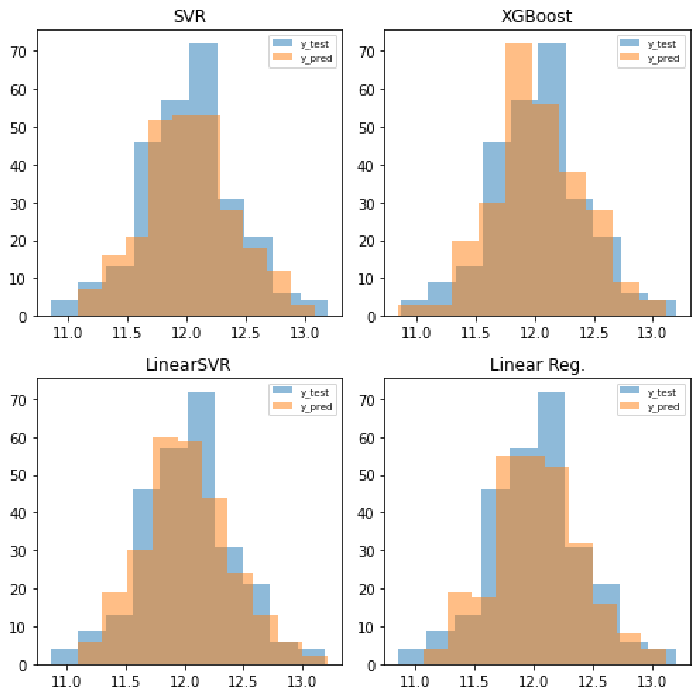


Figure 4.2: Sale Prices vs Predicted Sale Prices

4.3 Future Work

The future works on this project can include (i) studying other feature selection methods such as Random Forest feature importance, (ii) using other Boosting algorithms such as LightGBM, GBM, as XGBoost is in our best three models, we can assume that the mentioned boosting algorithms also can perform well.

Chapter 5

Appendices

5.1 Algorithms Implementation in Python

The GitHub Repository URL with our code:
https://github.com/TatevikJ/House-Price-Prediction.git
The data and .ipynb files are attached there.

Bibliography

[Tru+20] Quang Truong et al. “Housing Price Prediction via Improved Machine Learning Techniques”. In: *Procedia Computer Science* 174 (2020). 2019 International Conference on Identification, Information and Knowledge in the Internet of Things, pp. 433–442.

[Zho20] Yichen Zhou. “Housing Sale Price Prediction Using Machine Learning Algorithms”. In: (2020).

[HTW21] Winky K.O. Ho, Bo-Sin Tang, and Siu Wai Wong. “Predicting property prices with machine learning algorithms”. In: *Journal of Property Research* 38.1 (2021), pp. 48–70. eprint: https://doi.org/10.1080/09599916.2020.1832558.