Solving house price using different advanced techniques

Abstract

The problem that our group is trying to tackle is to forecast housing prices based on historical data, which falls under supervised learning. Three of us each employed a different method to achieve so: Neural Network by Zhongqi Yue, Ridge/Lasso regression and Ensemble Model by Chenkai Zhu, and Multiple/Polynomial regression by Zhengnan Zhu.

1 Introduction

Nowadays, more and more people are willing to buy a house instead of renting a house. Everyone wants to purchase the best house with the least amount of money. However, it is difficult for a home buyer to decide whether the stated price is overvalued or not. Therefore, it is crucial to estimate the price of the house accurately given the explanatory variables describing every aspect of residential homes.

We will use 79 explanatory variables to help with estimating the true value of a house such as MsZoning (Identifies the general zoning classification of the sale), Street (Type of road access to property) and Alley (Type of alley access to property), etc.

2 Data Cleaning and Preprocessing

The first thing to do is to convert data from raw data into model-ready data. There are different definitions of what is model-ready data, depending on the specific problem. In this paper with house price data, the following cleaning steps were executed. The first thing to do when cleaning any data is to either delete or impute missing values. There are different ways to impute missing values for numerical data. Some simple ways include computing the average, computing the mode or simply imputing with an appropriate constant like zero. In this dataset, many columns have too many missing values and imputing might not be beneficial to model building. So dropping columns with many missing values above a threshold is the most appropriate way.

Next, numerical and categorical data should be handled differently using different methods. Hence, the data is separated based on their data types in different columns. For numerical columns, the most important cleaning is to reduce the correlation among dependent variables. Hence, any columns with correlation larger than 0.8 are dropped. This is particularly useful in this house price dataset. This is because of the nature of house price data. Many features that affect house price are correlated, such as size of the house and size of the basement. For categorical variables, SelectKBest function was used to select the top 20 important variables, based on their p-values from statistical tests. Finally, these selected features will be combined together for model building.

3 Modelling

3.1 Solving regression problem with neural networks

The traditional way of solving regression problems is regression analysis (RA). In RA, The target is usually equated with the explanatory variables using the following equations:
$y_i = \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + \ldots + x_{ip}\beta_p$, where y_i is the target of subject i and x_{i1} to x_{ip} are the explanatory variables. In RA, goal is to try estimating the β parameters using different statistical likelihood like maximum likelihood estimation (MLE). However, there are lots of restrictions and limitations. For example, RA assumes independence among explanatory variables, which is not going to be true in many cases like the house price dataset used in this project. Usually larger houses have more bedrooms and more bathrooms, then these two explanatory variables are correlated. This is known as "multicollinearity", which leads to problems of analysis (Atici, 2011). Another limitation is that RA generally assumes strong correlation between dependent and independent variables (Atici, 2011). These problems can be reduced by neural networks.

Although neural networks (NN) still suffer from multicollinearity, they are much less sensitive compared to traditional RA. NN has the ability to learn the most relevant features and avoid learning redundant features through the use of hidden layers and activation functions. Furthermore, NN has more advanced techniques like regularization, dropout and early stopping to reduce the impact of multicollinearity. Secondly, NN has the ability to learn complex relations between independent and dependent variables, unlike RA that assumes strong correlation between them. This is mainly because of the hierarchical representation of NN. According to the universal approximation theorem, NN can approximate any continuous functions using only one single hidden layer with a finite number of neurons. Non-linear activation functions, larger number of parameters and hierarchical style of NN all contribute to these advantages that NN has.

3.1.1 Constructing Neural Networks

There are many different types of neural networks. The first one is feed forward ANN with back propagation (FFANN). The second type is cascade feed forward ANN (CFANN), in this type of ANN the inputs are used to feed the hidden layer. The last type is Elman ANN (EANN) (Khrisat & Alqadi, 2022). The most popular ANN is the back propagation ANN, which will be investigated in this section. Cascade feed forward ANN uses a cascade style, which means neurons are added to the network dynamically during training, to improve performance. Elman ANN is a special type of recurrent neural networks (RNN).

In feed forward ANN with back propagation (FFANN), there are two passes for each weights' update. First, the network computes a forward pass that will calculate weights and bias to make a "prediction" output. This output will be compared with the truth using the loss function specified. Then in the backward pass, the gradients of the error with respect to the weights at each level are propagated back. Finally, the network will use the propagated loss to make updates to the weights and biases. Note that heuristic approach should be applied to design the architecture of NN because proper numbers may even alter with different simulations in the same problem (Jang & Topal, 2013). This section of the paper will show understanding of neural networks with back propagation.

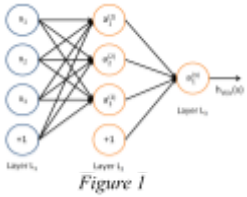


Figure 1

The forward pass can be represented with matrices. Let i and j be the number of neurons in layer m and $m + 1$. Then the weight at layer m can be represented with a matrix that has $i + 1$ number of rows and j columns. The first row is the weights for bias and entry (a, b) is the weight from neuron a in layer m to neuron b in layer $m + 1$. When doing the forward pass, the source, which is the matrix to be multiplied with the weights, needs to be specified. In the first input layer, the source should be the data, and in the consecutive hidden layers, the source is the output from the previous layer. Hence, weights can be represented by a three-dimensional matrix, that each 2D matrix is the weight for a single layer. In the forward pass, the source will dot product with the transpose of the first weight matrix. Then the output is passed as the source and fed into the next layer. For example, the below operation can represent the forward pass of the first layer of neural net in figure 1.

$$\begin{bmatrix} W_{01} & W_{02} & W_{03} \\ W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix}^T \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{11} & x_{21} & \dots & x_{n1} \\ x_{12} & x_{22} & \dots & x_{n2} \\ x_{13} & x_{23} & \dots & x_{n3} \end{bmatrix}$$

In backward pass, after computing loss using the loss function specified, the loss is propagated back to every layer in the network. Network takes derivative of the loss with respect to the weight at each layer to compute a δ_i value for each layer, representing the gradient of loss at layer i. However, due to chain rule, to compute the gradient in layer i, one must compute gradients at layer $i + 1$. So, we need to compute recursively from the last layer and propagate results to earlier layers.

Let a_i^l be the output of neuron i in layer l, z_i^l be the output after activation function g, and let n be the number of layers in total. In the last layer (base case), to compute gradient with respect to a_i^n, the network needs to compute gradient of loss with respect to z_i^n, and multiply it with g' due to chain rule. Then in recursive cases, the network can use δ_{i+1} from previous recursions to compute δ_i. δ_{i+1} has already included the gradient of loss up to a_{i+1}. Note that $a_{i+1} = g(a_i)W^{i+1}$. Hence, to compute gradient from a_i to a_{i+1}, the network uses chain rule again to multiply partial derivatives together. Hence, $\delta^i = \delta^{i+1}W^{i+1}g'(a^i)$.

The last missing piece to complete the entire picture is how to update the weights. There are many different optimizations the network can do to update its weights. Some famous ones include Adam, Adagrad and SGD optimizer. Since this section focuses on forward and backward pass in neural networks, optimizers will not be introduced in detail.

3.2 Solving regression problem with Ridge and Lasso Regression

My method is to use a linear model with Ridge and Lasso regression to see which regularization provides more accuracy. Later, combined with an XGBoost model, we will be able to take a weighted prediction across two different uncorrelated models.

3.2.1 Regression techniques

They are both popular regularization techniques for linear models that introduce a penalty term to prevent overfitting.

To further investigate this and see how they can be useful to our housing example case, we first take a step back and look at the basic linear model. The loss function for the basic linear model can be shown below.

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2$$

Then we would add a penalty term to the loss function to penalize the weights *w* to reduce model complexity. The loss function the Ridge regression is the following:

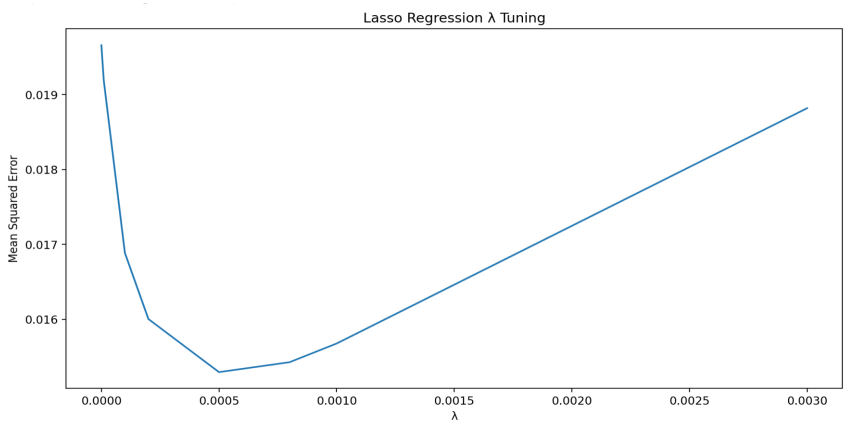
$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} w_j^2$$

The loss function the Lasso regression is the following:

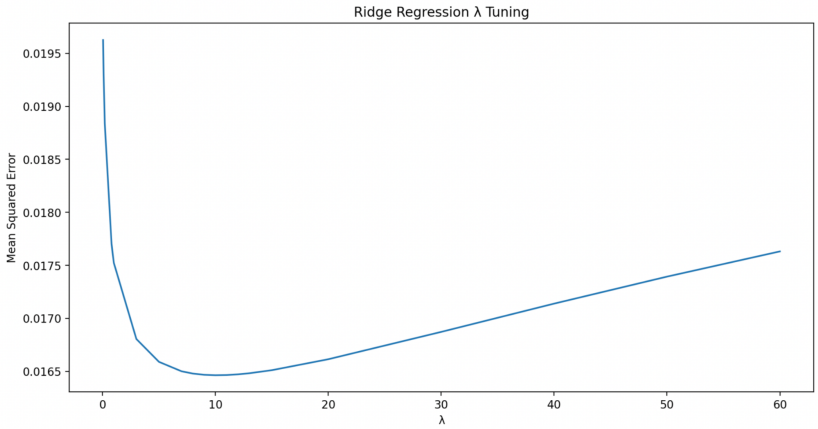
$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} |w_j|$$

Both techniques introduce a lambda term, λ, and it determines the amount of shrinkage that fixes the overfitting problem. The difference is that Lasso regression can also perform feature selection. When λ = 0, it means that all features are selected. When λ gets larger, it means more features get eliminated.

Therefore, λ will be our hyperparameter. According to a research paper published at International Conference on Information Technology and Nanotechnology, "cross-validation is the technique that can be used to find a 'proper' value for the λ parameter given a sample" (Melkumova, L.E. & Shatskikh, S.Ya., 2017).
We will apply Ridge and Lasso regression models with cross validation on our housing data by calling sklearn.model_selection.cross_val_score with 5-fold, and Mean Squared Error as the scoring function.



As we can see, when λ = 10, Ridge regression produces the smallest error.

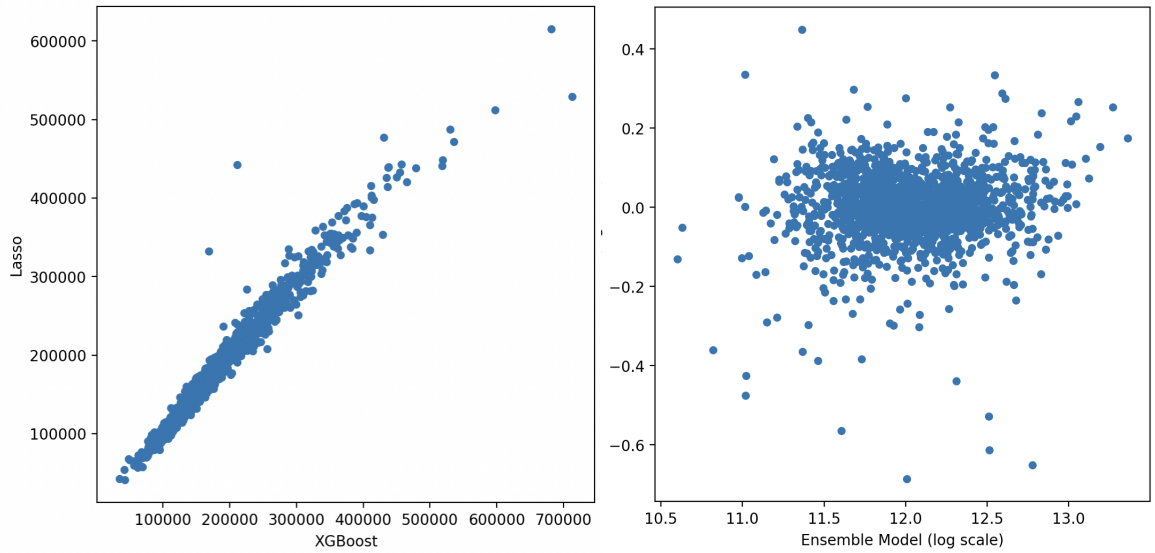


λ in Lasso forms an inverse relationship with Ridge. As we can see, when λ = 0.0005, Lasso regression produces the smallest error. When comparing the error, Lasso regression is able to attain an error value below 0.016. Therefore, in this case, Lasso is the winner.

3.2.2 Ensemble Model

The last thing is to give our Lasso model a bit of an extra accuracy boost is to use XGBoost. On the left it shows the prediction results between XGBoost and Lasso. It shows that both techniques produce very similar results, except for a few outliers.

According to a paper published by researchers at Iowa State University, "aggregating multiple learners through an ensemble of models aim to make better predictions by capturing the underlying distribution of the data more accurately" (Shahhosseini et al., 2022). We will take weighted combinations of each model of 70-30 split. From the diagram on the left, it shows that it achieves a good residual.



3.3 Solving regression problem with Multiple and Polynomial Regression

Multiple regression is a linear transformation of the X variables such that the sum of squared deviations of the observed and predicted Y is minimized. The prediction of Y is accomplished by the following equation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \varepsilon \ (1)$$

Where ε is an inconspicuous arbitrary variable with mean 0 and difference σ^2. The relationship portrayed from the equation above is known as a linear regression model, where β_i are the coefficient to help transform input variables to output. Those coefficients represent the independent contributions of each input variable to the estimation of the output variable.

3.3.1 Least squared error approach

The usual method to estimate the regression parameters by the method of least squares. This is an extension of the procedure used in simple linear regression. The sum of the squared errors is required to find a set of estimators that minimize the sum.

The equation (1) can be rewritten as the matrix form:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{e} \ (2)$$

We could get the error from equation (2)

$$\boldsymbol{e} = \boldsymbol{Y} - \boldsymbol{X\beta} \ (3)$$

Then, we will find estimator $\hat{\beta}$ to minimize the sum of squares of the errors

$$\boldsymbol{e^T e} = (\boldsymbol{Y} - \boldsymbol{X\beta})^T (\boldsymbol{Y} - \boldsymbol{X\beta})$$

After taking the derivative with respect to the vector β and multiply the inverse matrix of $(X^T X)^{-1}$, we get:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T Y}$$

Vector $\hat{\beta}$ is an unbiased estimator of β. The fitted predicted values for the mean of Y is computed by:

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\boldsymbol{X^T X})^{-1} \boldsymbol{X^T Y}$$

3.3.2 Polynomial Regression Model

Regression examination includes distinguishing the connection between a dependent variable and at least one independent variables. It is a standout amongst the most imperative statistical instruments which is widely utilized in all sciences. It can be expressed as:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 {X_2}^2 + \cdots + \beta_p {X_p}^p + \varepsilon = \beta_0 + \sum_{i=1}^{p} \beta_i {X_i}^i + \varepsilon$$

The mean squared error MSE is an unbiased estimator of the variance σ^2 of the random error term and is defined as:

$$MSE = \frac{SSE}{df_E} = \frac{\sum_{i=1}^{n} (y_i - \hat{y})^2}{n - (k+1)}$$

Where y_i are observed values and $\widehat{y_i}$ are the fitted values of the dependent variable Y for the i th case. Since the mean squared error is the average squared error, where averaging is done by dividing by the degrees of freedom, MSE is a measure of how well the regression fits the data.

3.3.3 Dummy Code for categorical variables

Since categorical predictor variables cannot be entered directly into a regression model and be meaningfully interpreted. We need to use other methods of dealing with information of this type. Dummy code would be a good choice to make a categorical variable with k levels into several variables with only two levels. In our case, “MsZoning”, “Alley”, “LotShape” and “LandContour” are all categorical variables that need to be eliminated.

4 Compare and contrast

Two advantages that neural networks have are their scalability and non-linearity. Neural networks are easy to scale up and down in a very flexible way. We could simply add or remove neurons and layers to make the model more complex or simpler. This gives much flexibility for neural networks to work with data of different sizes and problems of different kinds. Another advantage is its non-linearity. Although some other models can also handle non-linearity, neural networks can handle it more flexibly by adding or removing activation functions at each layer. One disadvantage is that neural networks are easy to overfit, and hence to reduce the robustness of the model. However, there are different methods for neural networks to overcome this issue, such as early stopping, dropout and so on.

The main advantage of my approach is to have an ensemble of two different non-correlated models: Lasso regression and XGBoost. It further improves the accuracy of the model. The advantage for Lasso regression is that it automatically performs feature selection for us through the λ value. Therefore, for some of the less important features, the weights are negligible. At the same time, Lasso gives more attention to the more important features. Therefore, it achieved a better result than Ridge. One area that could be improved through future iterations is to combine multiple regression models with different techniques including the ones introduced in this report. As a result, it could further increase the prediction accuracy.

The biggest advantage of using polynomial regression is that it provides an excellent approximation of the relationship between dependent and independent variables. A broad range of functions can be fit under it (almost all the regression problems). The disadvantage is that the presence of one or two outliers in the data can seriously affect the results of the nonlinear analysis, which means it is quite sensitive to the outliers. Therefore, it is significant to clear unnecessary data in the data cleaning phase.

5 Conclusion

In conclusion, neural networks did well in capturing relations in house price data, although neural networks are not the traditional way of doing regression analysis. By training neural networks, it learned complex features like size, location and was able to make accurate predictions on unseen data. However, there are still limitations of neural networks like overfitting. Also, the accuracy of the model will highly depend on the representation and quality of input data. Finally, the interpretability of neural networks is also limited compared to the parameters in transitional RA. But in this data specifically, neural networks did well in capturing the relationships and making predictions.

Regression analysis is a good statistical tool for the investigation of relationships between variables. The multiple regression analysis is a useful method for generating mathematical models where there are several (more than two) variables involved. The Lasso regression achieved exceptional results on a large set of features, since it automatically performs feature selection through its lambda value. Polynomial regression model consists of successive power terms. Both have shown sufficient accuracy for the prediction of house price. On the top of that, combined together as an ensemble further improve the prediction accuracy.

Reference

Atici, U. (2011). Prediction of the strength of mineral admixture concrete using multivariable regression analysis and an artificial neural network. *Expert Systems with Applications*, *38*(8), 9609–9618. https://doi.org/10.1016/j.eswa.2011.01.156

Khrisat, M. S., & Alqadi, Z. A. (2022). Solving multiple linear regression problem using artificial neural network. *International Journal of Electrical and Computer Engineering (IJECE)*, *12*(1), 770. https://doi.org/10.11591/ijece.v12i1.pp770-775

Jang, H., & Topal, E. (2013). Optimizing overbreak prediction based on geological parameters comparing multiple regression analysis and artificial neural network. *Tunnelling and Underground Space Technology*, *38*, 161–169. https://doi.org/10.1016/j.tust.2013.06.003

Ostertagová, E. (2012, November 2). *Modelling using polynomial regression*. Procedia Engineering. Retrieved April 13, 2023, from https://www.sciencedirect.com/science/article/pii/S1877705812046085

Kumari, P., & Kumari, A. (n.d.). *A Study of Polynomial Regression towards Machine Learning*. A study of polynomial regression towards machine learning - ignited minds journals. Retrieved April 13, 2023, from http://ignited.in/p/58015

Shahhosseini et al. (2022). *Optimizing Ensemble Weights and hyperparameters of machine learning models for regression problems*. Machine Learning with Applications. Retrieved April 13, 2023, from https://www.sciencedirect.com/science/article/pii/S2666827022000020

Melkumova, L.E. & Shatskikh, S.Ya. (2017). Comparing Ridge and LASSO estimators for data analysis. Procedia Engineering. 201. 746-755. 10.1016/j.proeng.2017.09.615.