Predicting the Ames, Iowa, Real Estate Price

Author: Zhongyuan Ye(zy2550), Sarah Kim(sk4541), Chengyi Gong(cg3355), Lucy Zhang(xz3141), Ziyu Liu(zl3220)

Introduction

Having a clear understanding of the real estate market, especially its prices, will help buyers make more strategic decisions when purchasing a property. In this study, we examine the relationship between a property's price and its various features, such as number of bedrooms, total area, etc. We utilized the dataset containing 1460 Ames, Iowa, house price records from Kaggle. Through data exploration and visualization, we were able to discover the basic relationships between the price and various independent variables. We then deployed machine learning techniques, including linear regression, tree-based models, and neural network model, to predict the prices given the features. At last, we found the best machine learning model for the house price prediction task and the top 10 features that contribute to the house price.

Data Preprocessing

Originally the dataset had 81 columns (38 numerical and 43 categorical) and 1460 rows. One of the columns, 'SalePrice', served as the target variable. While the original target values (house price) were skewed to the right slightly, the logs of target values were normally distributed, which indicated that we don't need to sample for the regression modeling. 10 of the original 81 columns were dropped due to a high percentage of missing rows (>= 80.8%) or high correlation with another feature (correlation >= .8). We also encoded the 43 categorical columns in the dataset to become numerical ones– 9 were mapped using ordinal encoding, 14 using one-hot encoding, and 20 using target encoding. After preprocessing, the raw dataset was changed to a preprocessed dataset with 127 columns and 1445 rows. In the following model below, we split the preprocessed dataset into the development and test dataset using an 80/20 split.

Model Development

Based on the data exploration and visualization, we noticed that there is a potential linear relationship between the property's price and its various characteristics (e.g. number of bathrooms, total area, etc.). Therefore, we deployed linear regression and its advanced forms for an initial prediction. To make the prediction more accurate, we also used tree-based methods and neural networks, and then determined the most appropriate model to use for this case based on model performance.

Regression Model

Ridge and Lasso Regression:

For linear regression, we deployed Ridge and Lasso regression to prevent the potential overfitting problem. In Ridge regression, the parameters are tuned within the range of 10 logarithmically spaced values between 0.001 and 1000. The best parameter results are 215.44, with R^2 being 0.8744 and MSE being 0.1436. In Lasso regression, the parameters are tuned within the range of 30 logarithmically spaced values between 0.0001 and 100. The best parameter results are 0.0189, with R^2 being 0.8795 and MSE being 0.1378.

Gaussian Process Regression:
Using Gaussian Process Regression (GPR), we tuned the hyperparameters "kernel", "alpha", and "n_restarts_optimizer" using grid search and cross-validation with 5 folds. The optimized model achieved a satisfactory R-squared score of 0.8301, indicating a good fit between GPR and the data. The MSE of 0.1943 indicates that, on average, GPR's predictions deviate from the actual values by 0.194 units. Furthermore, the RMSE of 0.2591 indicates that the average deviation between predicted and actual values is 0.259 units. During the optimization process, we encountered a Convergence Warning that persisted despite changing the kernel, optimizer, and scaling the dataset. This suggests that GPR may not be the best model for our dataset, despite the reasonably good metrics achieved. In conclusion, our GPR model performed well on the dataset, but given the Convergence Warning, it may be worth considering other models as well.

Tree Based Model

From the linear regression model performance, we noticed that there might be a potential overfitting problem. Therefore, we also deployed tree-based models for more accurate results. Specifically, we developed Random Forest and XGBoost.

Random Forest:
In Random Forest, we used grid search as the technique to tune hyperparameters. In detail, the number of estimators was tuned within the range (25, 50, 75, 100), and the maximum features were tuned within the range of (10, 15, 20, 25). Through the tuning process, we found that 15 features and 25 estimators give the best model performance, with R^2 equal to 0.8776 and MSE equal to 0.1399.

XGBoost Model:
For XGBoost model, we used grid search to tune hyperparameters, including the number of estimators, learning rate, and max depth. Then we plot the feature importance from the XGboost model, which includes all the features. Then we first trained an XGboost model with the best hyperparameters selected by the grid search and made predictions based on the test data. The result shows that The Mean Squared Error (MSE) is 0.1067, and the R^2 Score is 0.9066. Then we trained another model which only included the top 50 important features and got 0.1054 for the Mean Squared Error (MSE) and 0.9078 for the R^2 Score. Comparing the two models based

on the R^2 score, the second model, which included only the top 50 important features, turned out to have the performance.

Neural Network

For the Neural Network model, we tuned the hyperparameters of the Keras sequential model with 3 hidden layers and 1 output layer: the first layer has 64 layer size and uses Relu for activation; the second layer uses 32 layer size and uses Relu for activation; the third layer uses 16 layer size and uses Relu for activation, the final layer (output layer) has 1 layer size and use linear for activation (regression problem). Hyperparameters tuned in this model are the learning rate of the optimizer ([0.001, 0.005, 0.1]) and epochs([10, 25, 50]). Our team split the development dataset into 'train' and 'validation' sets for this model with an 80-20 ratio, trained each model with the training dataset, and recorded each epoch's performance on predicting the validation dataset.(figure 1)

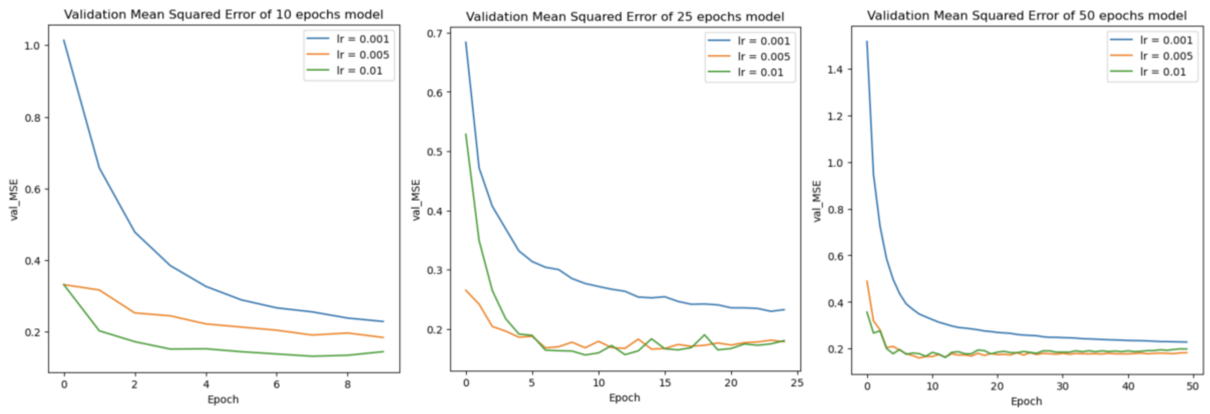


Figure 1: epochs-based validation set MSE of the neural network model

From Figure 1 above, when training each model with 10 epochs, the high learning rate model will result in better performance. This might be because the epochs number is too low, which results in all three models being underfitted. When training each model with 25 epochs, a learning rate of 0.005 has the best result among all models: the validation MSE of the model with 0.01 learning rate bounces each epoch because learning rates are too high, which leads to divergent behaviors; the validation MSE of the model with 0.001 learning rate keeps decreasing which means the model is still underfitting. When training each model with 50 epochs, the validation MSE of both the 0.005 and 0.01 learning rate models are increased compared to the '25 epochs' models because of overfitting, and the validation MSE of the 0.001 learning rate model keeps decreasing, which means this model is still underfitting.

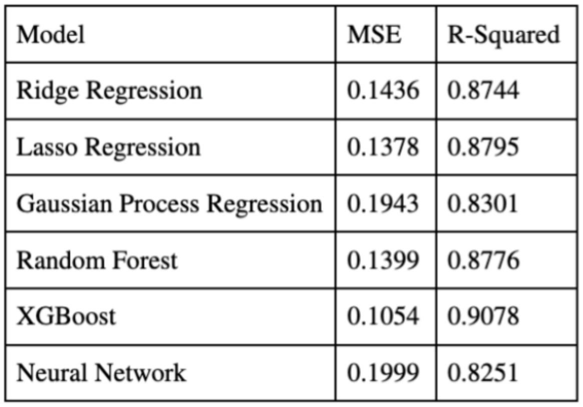
In conclusion, the optimal hyperparameters for training are 0.005 learning rate and 25 epochs. The test MSE is 0.1999, and the test R^2 is 0.8251 for the Neural Network model with a 0.005 learning rate and 25 epochs.

Conclusion:

From Table 1 below, we found that the XGBoost model trained with the 50 most important features performs the best among all other models on predicting test set housing price (has the lowest MSE and R-Squared), which means that the XGBoost model is the fittest model for house price prediction task.

Our team also found that the non-linear model like Random Forest or Neural Network has relatively bad performance compared to other linear model like XGBoosting or Lasso Regression. Hence, we prove our assumption and conclude that the relationship between housing price and each column is a linear relationship.

What's more, from the feature importance figure of the XGBoost (figure 2 below), our team found that the top 10 most high-weight columns during housing price prediction are 'OverallQual,' 'BsmtQual_5.0', 'Neighborhood,' 'CentralAir_Y,' '2ndFlrSF', '1stFlrSF', 'KitchenQual_5', 'FullBath,' 'GarageType,' 'ExterQual_5'. From this, we infer that the top 10 important features that determine the house price are 'overall material and finish of the house,' 'the height of the basement,' 'Neighborhood,' 'Central air conditioning,' 'Second-floor square feet,' 'First-floor square feet,' 'Kitchen quality,' 'Full bathrooms above grade,' 'GarageType,' and 'Quality of the material on the exterior.' We believe our conclusion can help the future house buyer and seller better estimate the house price and can help the future house designer understand how to increase the house value when building the house.



Model	MSE	R-Squared
Ridge Regression	0.1436	0.8744
Lasso Regression	0.1378	0.8795
Gaussian Process Regression	0.1943	0.8301
Random Forest	0.1399	0.8776
XGBoost	0.1054	0.9078
Neural Network	0.1999	0.8251

Table 1: Test MSE and R-Squared for each model

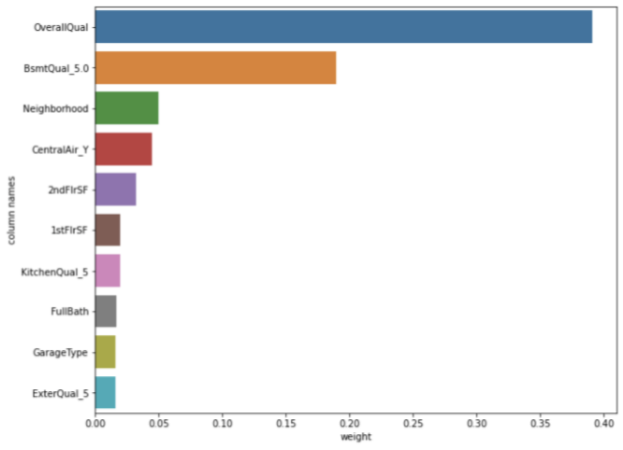


Figure 2: 10 most important column in XGBoost model