Solving Regression Problems Using Classic ML Algorithms*

Bang, Ashley
Department of Computing and Informatics
University of North Carolina at Charlotte
Charlotte, NC
sbang6@charlotte.edu

***Abstract*—Regression problems are present all around us, where predicting the correlation and relationship of certain variables leading to specific outcomes is the basis of a lot of research and leads to understanding. Regression based machine learning models are now commonly used to deduce the solution to such correlation problems and to form predictions. To gain a better understanding of how to utilize machine learning to solve regression problems, the AMES housing dataset is used with the goal of finding the best performing model to accurately predict housing prices. Five different models are implemented and compared to realize the most effective model for this housing prediction is XGBoost and standard generalization.**

I. Introduction

Predicting a continuous numerical value like housing prices, is not only difficult without machine learning, it is also important to discern which model to make use of. The real-world regression problem that will be covered is the famous AMES housing dataset taken from real homes in Iowa [1]. The goal is to gain greater understanding of the process in which machine learning models are applied and deduce the most accurate, best performing model for this application. To understand the problem, the dataset must be understood. The AMES dataset used and implemented throughout this report contains about 2920 records and 80 feature categories. The feature variables include the typical attributes of home buying and filtering with the number of bedrooms, bathrooms, lot space, year built, square footage, etc. Even more in depth, the variables even cover home functionality rating, height of the basement, interior of the garage, etc [1]. Just by looking at the data and all the very specific and seemingly obscure features it is apparent the prediction cannot be made efficiently manually and requires a more robust algorithm. These features have some sort of relationship with the target SalePrice and it is the model's job to find it. Given this large list of features with varying levels of influence on the actual cost of a home, pre-processing takes place to better represent the data.

II. Methodology

A. Preprocessing and feature engineering

In order for data to be taken into our models, it is essential for them to undergo pre-processing. All features are not made the same and many can be omitted or modified in a way that is more digestable to the model. First, all the values in the columns can be filtered through with a simple percentage metric to see the amount of null values in a given column or feature variable. Those with more than 70 percent of its values missing can be dropped from the dataset as they are not significant to the overall problem. This is a very simple method of feature cleaning that gets rid of unnecessary values that would only cause greater load on the model. Further, pre-processing pipelines are implemented for a cleaner implementation of feature engineering. A simple imputer pipeline call is used to replace the remaining missing values with estimated values. The value used in this report's models is the median as models cannot handle missing values and the median value would not cause significant skewing. There are other strategies outside of using the median, such as mean, mode, and more advanced techniques like iterative imputation. As previously mentioned, there are not only numerical variables but also categorical variables that require proper numerical labeling. One-hot encoding is used to achieve this as it is a reliable way to encode by assign a constant value one to a given category and zero to all the others. This prevents ordinal bias or false-ordering bias from occurring, which falsely places more or less weight or bias on larger and smaller assigned numbers, respectively. Standardization can be implemented to further reduce bias by transforming features to have a mean of zero and standard deviation of 1. This is not the most essential for this dataset but it is a helpful feature engineering step for cases where numerical stability is important. These pre-processing methods can be further customized to be used for specific columns and are not restricted to being applied to the whole set. ColumnTransformer is used to selectively apply pipelines to columns. Standardization was applied to numerical features and one-hot encoding was applied to categorical features with this. For one model, binning was applied to continuous numerical variables to convert it to be used as a discrete category. This was the extent of feature engineering for the selected models that will be covered in the next section. There are far more ways to engineer the features in a way that would yield greater results. Other methods that were not applied include normalization, target encoding, log transformation, and dimensionality reduction.

B. Explanation of the classical models used

The three classic models that were covered in class and also used for this problem include: linear regression, neural network, and naive Bayes. Linear regression will be used as our control model as it's simple yet highly applicable for this regression problem with continuous numeric values.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \tag{1}$$

$$J(\boldsymbol{\beta}) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2}$$

$$J(\boldsymbol{\beta}) = \frac{1}{2n} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{3}$$

This model is fast and easily interpretable, as it assumes linear relationships between features and targets, which provides a good baseline for comparing with the performance of the other models. Neural network algorithms also use regression, but it is often used in complex interactions because of its ability to handle non-linear relationships and larger load.

$$z = \mathbf{w}^T \mathbf{x} + b \tag{4}$$

$$\mathbf{a}^{(l)} = \phi\left(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{5}$$

We can compare the performance of the neural network model with the linear regression model to see how well it can predict based on the complexity and how the process of reaching a higher performance differs (harder to tune that XGBoost and easier to overfit without regularization). The third model used from this course is naive Bayes, a classification model.

$$P(C_k \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_k)\, P(C_k)}{P(\mathbf{x})} \tag{6}$$

This classification model does not belong in this regression problem as it used for discrete categorical features. It assumes feature independence which holds the opposite goal of trying to understand the feature relationships with the housing price. However, this model was included to convey the importance of understanding the problem in order to define the right model to solve the problem. For this application, the model will implement binning (loses info) the SalesPrice into categories: "Low", "Med", and "High", to essentially convert this into a classification problem. This more closely mimics the goal of prediction without actually predicting housing prices but rather placing them by affordability. Though this might be helpful for some applications, it is not the goal of this assignment nor the solution to the problem, highlighting why the understanding of the problem and appropriate model is required.

C. Summary and implementation of the two papers

In addition to the three proposed models are two more recent methods and algorithms used for regression cases. In 2024, Sharma et. al compared the performance of XGBoost(extreme gradient boosting), support vector regressor, random forest regressor, multilayer perceptron, and multiple linear regression algorithms in predicting housing price using the same AMES dataset [2]. The paper explains in detail the background and usage of each algorithm, providing the logic behind them before going in depth about the implementation. Similar steps are followed as mentioned in the the pre-processing and feature engineering with the inclusion of model tuning that takes place after training. The evaluation metrics used to represent the models' performance include R-squared, MSE, and MAE. The results of comparing the algorithms conveyed XGBoost provided the best evaluation metrics. IBM defines XGBoost as an, "open-source machine learning library that uses gradient boosted decision trees, a supervised learning boosting algorithm that makes use of gradient descent. It is known for its speed, efficiency and ability to scale well with large datasets" [3].

$$\mathcal{L} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{7}$$

This is strong because decision trees split non-linearly, boding well for non-linear relationships as this housing problem contains. XGBoost is the regression model replicated from the paper's existing open code to represent the first paper of two. The paper proposed by Krishnan et. al in 2020 explored more improved methods of utilizing existing machine learning models [4]. The models that are compared in this paper are XGBoost, hybrid regression, and stacked generalization. The latter two methods both use multiple regression model but the difference lie in how they are structured. Hybrid regression models takes two or more regression models as a combination whereas stacked generalization occurs as an ensembling technique with layers of models where predictions from previous layers act as the features for the next layer's model. The specific technique used in this paper stacked 2 layers of architecture where the first layer uses RandomForest and LightGBM (Light Gradient Boosting Machine) and the second layer, XGBoost. LightGBM like XGBoost also uses gradient descent but tends to work faster and be more memory efficient while not providing as much stability as XGBoost. This layering architecture was replicated for the purpose of this report as the representative algorithm for the second paper. The implemented code was not drawn from the paper itself but simply followed the structure to test the performance in a more controlled manner in comparison with the other algorithms.

III. Results and Evaluation

First, a simple histogram showing the classification model of the group is shown below. It have completely different evaluation metrics, making it more difficult to numerically deduce and compare with the other classes yet we observe an accuracy score of 0.53 [5]. This is widely considered to be quite low as some information is also lost in the binning process in conjunction with the fact that the housing prediction task is a regression problem. This model can be deduced as the worst performing for these reasons.

The following table displays the two evaluation metrics found across each of the regression models. (The evaluation metrics for the naive Bayes model is separate as there is no loss function so that will be covered later.)

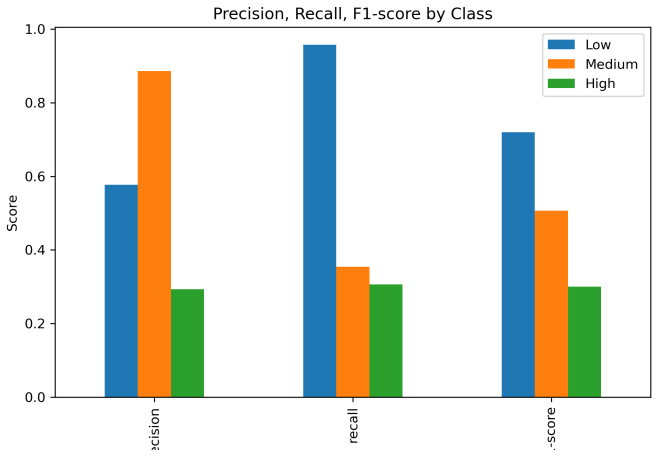

Fig. 1. Naive Bayes Precision and Recall

TABLE I
MSE AND R^2 SCORE OF REGRESSION MODELS ON THE AMES HOUSING DATASET

Model	Mean Squared Error (MSE)	R^2 Score
Linear Regression	810,584,690.08	0.89
XGBoost	729,342,720.0	0.90
Neural Network	1,077,446,457.61	0.86
Stacked Generalization	754,781,952.00	0.90

Based on the values seen in table I, the XGBoost mod has the highest and best R-squared score as well as the lowe MSE, closely followed by stacked generalization. Then i table II we see slightly different results where XGBoost yiel the lowest RMSE value and stacked generalization yieldir the lowest MAE. These results suggest XGBoost and stacke generalization as the better performing models that best f this regression task within similar proximity to one anothe This can be explained by the nature of the stacked mod incorporating XGBoost in its second layer and the robustne of XGBoost itself as a non-linear model that still maintai its simplicity in comparison to neural networks. For furthe visualization, figures showing the correlation of actual and predicted scores for their respsective models can be found in this github link and the figures below: []b5

From the figures you can see the prediction points are more focused and centered around the actual line for XGBoost and Stacked Generalization while there are more predictions that stray for linear regression and most for the neural network model.

TABLE II
RMSE AND MAE OF REGRESSION MODELS ON THE AMES HOUSING DATASET

Model	Root MSE (RMSE)	Mean Absolute Error (MAE)
Linear Regression	28,470.77	17,351.37
XGBoost	27,014.16	16,860.79
Neural Network	32,824.4	19,063.86
Stacked Generalization	27,469.67	16,557.78

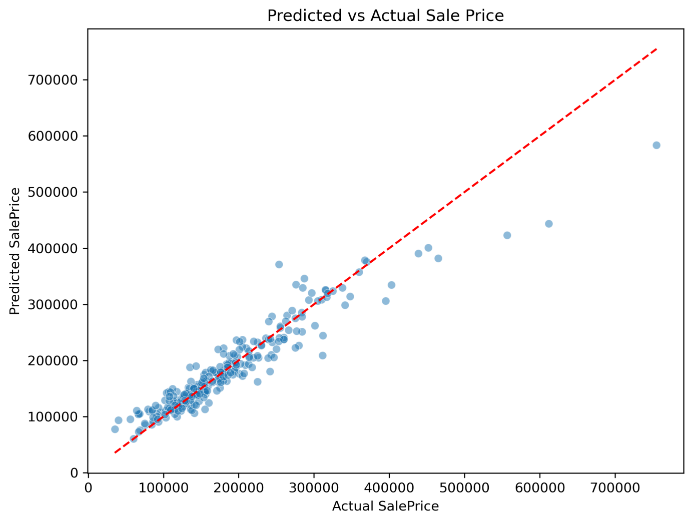

Fig. 2. Linear Regression Actual vs. Prediction

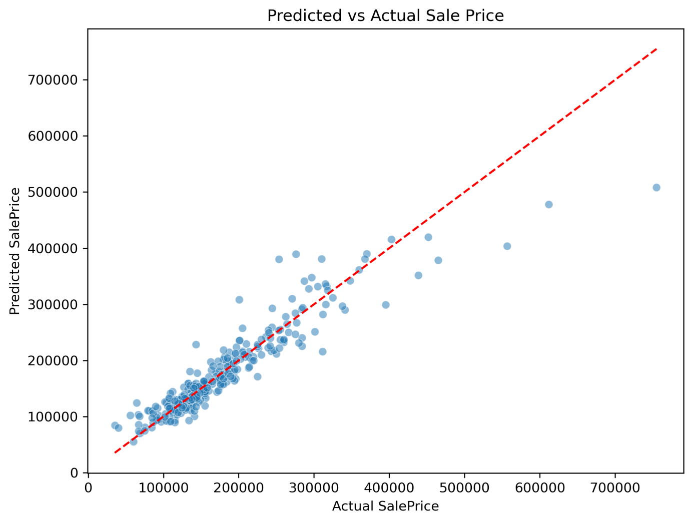

Fig. 3. XGBoost Actual vs. Prediction

IV. DISCUSSION

A. Model Performance

- Linear regression serves a good basis for this kind of regression tasks for its simplicity and interpretability and performed reasonably well with an R² around 0.89 but was not the best performing. The relationship between input features and the target can be directly understood while training quickly using minimal computational resources. It works best when the relationship between the predictors and target is roughly linear and the data

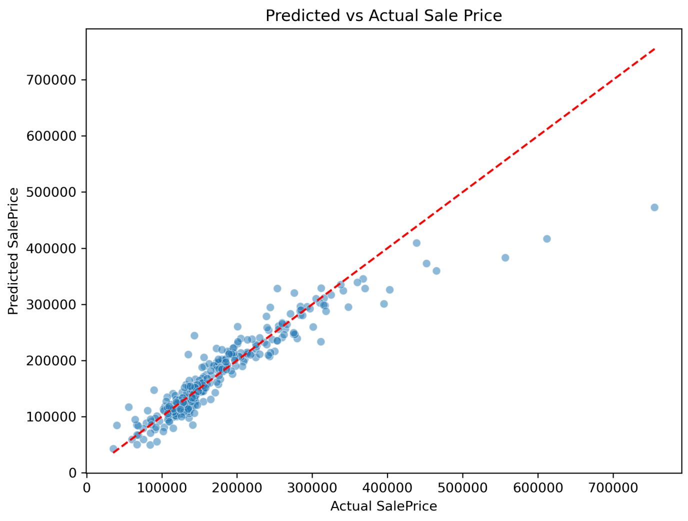

Fig. 4. Neural Network Actual vs. Prediction

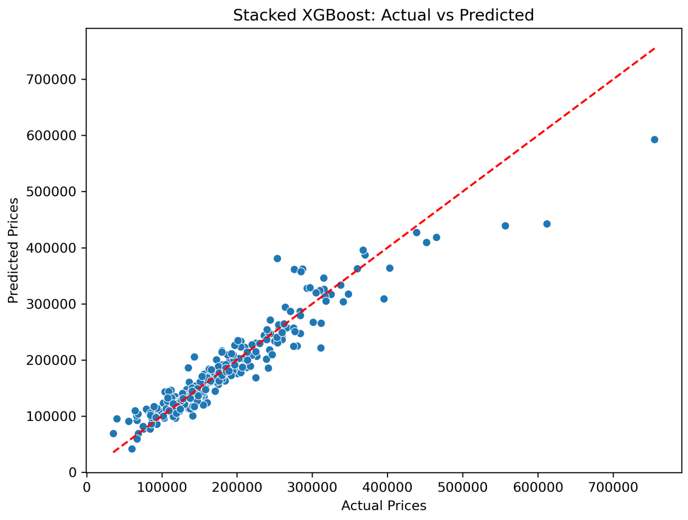

Fig. 5. Stacked Generalization Actual vs. Prediction

doesn't contain too many interactions or non-linearities. For this reason, it does not hold up as well because of the complexities that lie in the interactions that exist within the features and the non-linear relationships present.

- Neural networks can model nonlinear relationships, so they can capture interactions in housing features that linear regression cannot. However, the performance of this model is more sensitive to hyperparameters and pre-processing. Neural networks also require larger amounts of data to generalize well whereas the Ames Housing has only 1,460 observations, which is relatively small. For naive Bayes, due to its being primarily used in classification problems with its assumed conditional independence, it can't account for the complex dependencies in this task.
- XGBoost is a highly effective gradient boosting framework that handles non-linearity, feature interactions, and missing values efficiently that includes L1 or L2 regularization to prevent overfitting. It's fast and scalable, with optimized implementations that can handle large datasets with the trade off that the complexity may make it less interpretable than linear models. This means it requires careful hyperparameter tuning for optimal performance, but not as must as with neural networks.
- Stacked generalization combines predictions from the two base models to reduce individual model biases and variances. The meta-model, or the later layer XGBoost, works to correct errors from the first-level models, improving overall accuracy. This usually achieves higher R^2 and lower error metrics than any single model. The trade-off is how complex it is in comparison to just XGBoost or linear regression and how computationally intensive it is to train and validate. The interpretability decreases as layers of models increase and requires careful cross-validation to avoid data leakage between stacking levels.

B. *Challenges*

When reproducing the five models, one of the challenge faced during the process was with hyperparameter tuning and building a decent pre-processing pipeline. Deciding to go with the simpler route of building a somewhat standard pre-processing pipeline for each of the five models helped in simplifying the code but came with the fact that the models were not all optimized to the fullest. Implementing more with normalization, regularizing, binning, and more fine feature engineering and post-processing, the performance could have increased. The most difficult aspect of reproducing the models were for the more complex XGBoost and standard generalization. Not only are they new models to my knowledge base, they also require more tuning in ways that were not fully captured through my implementation. The results show the two models being quite comparable, however in the real world and in the Krishnan [4], standard generalization performed noticeably better. The paper used 5-fold cross validation which is useful in find the bias-variance trade-off, something that was not included in my reproduction. Attempting to fully replicate the more complex models without prior experience or in depth knowledge proved to be the greatest challenge that still allowed me to learn as I progressed through the project.

V. Conclusion

Housing prediction with the AMES dataset can be efficiently achieved using the right model and techniques that find a good equilibrium of complexity and interpretability. Understanding the dataset is the first important step of solving regression problems. This leads the direction for model selection. A regression problem with non-linearity and complex dependent relationships between features and target will require a model that can handle it and is suitable for training. There is also great merit in staying informed about topics within regression modeling where newer methods may outperform already commonly used models. Regression models can be incorporated with or in conjunction with one another to form more robust models and to offset the trade-offs of individual models when used in tandem. XGBoost and standard generalization using XGBoost in its stacking work the best out of the five discussed models. The model handles the bulk of the workload but the pre-processing and tuning are key aspects to a working algorithm. Understanding the effect of each hyperparameter on the model is required to tune efficiently. Future works for housing predictions can be done in the type of data that is used. All of the data in the AMES set were only numerical and categorical. However, there are many visual factors that go into the price prediction of a home and in other regression tasks. The housing market is heavily reliant on visual aids, so subsequent research that takes this into mind would prove to be useful not only for housing price predictions but for many others.

Acknowledgment

Warm regards and gratitude to Professor Xiang Zhang for never failing to emphasize the gift of learning and prioritizing this above all throughout the entirety of the course. For sharing your hard-earned expertise in a simplified way to us, thank you and I hope to have the pleasure of being taught by you again. To Yupeng Yang, thank you for your patience and rigor in grading and answering so many questions in a careful, timely manner. I assume this is highly unprofessional but given this is not a legitimate publication, I hope this brightens your time while grading.

References

[1] S. M. Zain, “House Price Prediction,” Kaggle Notebook, 2023. Available: https://www.kaggle.com/code/syedmuhmmadzain/house-price-prediction-ipynb/notebook

[2] H. Sharma, H. Harsora and B. Ogunleye, “An Optimal House Price Prediction Algorithm: XGBoost,” arXiv preprint arXiv:2402.04082, Feb. 2024.

[3] “What is XGBoost?,” IBM, 2025. [Online]. Available: https://www.ibm.com/think/topics/xgboost. Accessed: 03-Dec-2025.

[4] M. N. S. Krishnan, V. Saranya, and K. S. R. Saran, “Housing Price Prediction via Improved Machine Learning Techniques,” Procedia Computer Science, vol. 174, pp. 433–442, 2020.

[5] A. S. Bang, “IntroMachineLearningCapstone” (GitHub repository), GitHub, 2025. [Online]. Available: https://github.com/asbang/IntroMachineLearningCapstone. Accessed: 04-Dec-2025.