DABN14

Fall semester 2024

LUNDS UNIVERSITET

Project 1: Advanced Machine learning

Are regression Trees more than intuitive visualizers?

Regression Trees, Gradient boosting & Random Forest

Felix Hult

Evagelos Ifantidis

1. Introduction

There are many choices available for data scientists when faced with the choice of a machine learning model for any classification or regression problem. The choices range from simple models such as linear regression to more complex such as neural networks. Intuitively, these models can be hard to understand and interpret. Classification and regression trees are however known for being easily interpreted and visualized, making them an enticing choice when communicating data generated insight. Despite this, classification and regression trees are generally recognized as having a lower predictive power than other models (James, Witten, Hastie, Tibshirani & Taylor, 2023).

There have however been several proposed ways to fix this problem and increase the predictive power of classification and regression trees. This paper will explore these proposed "fixes" with the intention to answer the questions whether classification and regression trees also can be an effective predictive model in a regression setting outside of just providing easy to understand results and visualizations. This will be done in a regression setting where the goal is to predict sales prices of houses.

2. Literature Review

Classification and regression trees are grown by repeatedly splitting data into smaller and more homogeneous groups which are either characterized by the typical value of the response variable in that group (De'ath & Fabricius, 2000). The predictions are then either the majority vote (for classification) or the mean value (for regression). The trees are then easy to visualize and intuitive to understand and can be used for exploratory analysis. Moreover, there are more advantages such as the input variables can take any form, the variables used are unaffected by transformations and different scales and when the tree is being built, irrelevant predictors are rarely selected (Elith, Leathwick & Hastie, 2008). Trees can also be applied to data that have a

large number of cases as well as a large number of variables while being resistant to outliers (Sutton, 2005). However, small changes in the data can impact the tree structure and lead to different splits (Elith, Leathwick & Hastie, 2008) making it prone to overfitting.

As mentioned, methods for dealing with the inherent problems with classification and regression trees have been proposed, such as ensemble methods. These methods take the majority vote or mean of an *ensemble* of predictors (Loh, 2014). *Bagging* is such a method where bootstrap samples are drawn to create new trees. *Random forest* tries to de-correlate the trees by using different features for each split in the trees. *Boosting* is a method where trees are grown sequentially where each new tree puts more weight on the observations misclassified in the previous tree (Loh, 2014). The individual trees in the ensemble are called *weak learners* due to their mediocre predictive power on their own. But when combining the weak learners it is possible to achieve a very powerful model (James, Witten, Hastie, Tibshirani & Taylor, 2023). There are many more options available, but this paper will focus on the three methods mentioned above.

3. Method

3.1 Data

The dataset used in this paper is the *Ames Iowa Housing Data*, created by Marco Palermo (2020) and retrieved from Kaggle. The dataset was originally compiled by Dean De Cock (2011) for use in data science education. It includes a comprehensive set of predictors for the sale price of houses in Iowa. The dataset contains 79 explanatory variables and 2930 observations. Among these variables, 23 are nominal, 23 are ordinal, 14 are discrete, and 20 are continuous. The target variable for prediction is the *Sales Price* (see Table 1 for descriptive statistics). Data preparation was necessary to handle features with a significant proportion of missing values. A threshold of 10% missing data was implemented, meaning any feature with more than 293 missing values was removed from the dataset. Additionally, we ensured that all variables were assigned their proper data types. To improve interpretability, we transformed variables related to the year

houses were built, the year houses were remodeled, and the year garages were built. These values were converted into new features that represent the age of the house or garage at the time of sale by subtracting the respective year from the year the house was sold. The choice of this dataset was that it is similar to the common *boston housing* dataset but this set contains more features, which makes it an interesting choice since trees are known to be able to handle many features.

Table 1: Descriptive statistics of the target variable, *Sales Price*.

Metric	**Value**
Count	2,678
Mean	186,405.73
Standard Deviation (Std)	79,501.57
Minimum (Min)	12,789
25th Percentile (25%)	134,575
Median (50%)	166,650
75th Percentile (75%)	218,641.75
Maximum (Max)	755,000

3.2 Method

3.2.1 Baseline Model: Single Regression Tree with Pruning

As a baseline model to compare against more advanced methods, a single regression tree was implemented. An essential part of building trees is pruning (Kuhn, Page, Ward & Worrall-Carter, 2014). This process limits the tree from growing too large, potentially creating almost as many terminal nodes as observations. Pruning helps prevent overfitting by simplifying the tree structure, making it more generalizable to unseen data. The pruning process involves first growing a fully expanded tree and then systematically reducing its size through methods like *weakest link pruning* or *cost-complexity pruning* (James et al., 2023). This introduces a penalty

for excessive terminal nodes, controlled by a hyperparameter, which is typically optimized using cross-validation.

3.2.2 Bagging

As mentioned in the introduction,one drawback of tree-based methods is that they are very dependent on the input data and predictions can therefore vary with changes in the input data. Bagging is one ensemble method we can deploy to get better predictions. Bagging simply means that we create many different trees from new bootstrap samples (James et al., 2023). By averaging many sets of observations, we can reduce the variance and get more stable predictions.

In our case, we applied the bagging method and set the number of trees to be used in the ensemble to 5000.

One very nice outcome of doing the bagging method is that we can get a clear picture of which variables carry the most importance. We look at the total recorded amount that RSS decreased due to a split over a specific predictor and then we average it over all trees. This gives us insight into which predictors contribute the most to our model (James et al., 2023).

3.2.3 Random Forest

Our next method is Random forest. This is similar to bagging in that we produce many trees from bootstrap samples, however, at each split the model is not allowed to use all the predictors, rather only a small subset. This is done to decorrelate the trees and remove any effects of predictors that carry a lot of importance and will hence be the top split in every tree (James et al., 2023).

Determining the amount of predictors (m) that our trees are allowed to use at each split ultimately have an impact on our predictions, and hence our accuracy. A typical choice is $\sqrt{p}$. We performed a cross validation of some values we considered to be applicable and found that p/2 was the best parameter value for our problem, as seen in the plot below.

3.2.4 Gradient Boosting & XGBoost

Gradient Boosting is an ensemble learning method used to build predictive models by sequentially combining weak learners, typically decision trees. Each tree attempts to correct the errors of its predecessor by minimizing a loss function (e.g., Mean Squared Error for regression). The method relies on boosting, where the model emphasizes the training samples that are harder to predict. Gradient Boosting is particularly effective for capturing non-linear relationships in data and is known for its flexibility in handling both regression and classification tasks.

XGBoost (Extreme Gradient Boosting) is an advanced implementation of the Gradient Boosting algorithm, designed to optimize speed and performance. It incorporates regularization techniques (L1 and L2) to prevent overfitting and uses features like tree pruning, weighted quantile sketch, and parallelized tree construction to improve efficiency. XGBoost is highly scalable and efficient, making it one of the most widely used algorithms in data science competitions and real-world applications (Chen & Guestrin, 2016).

3.2.5 Evaluation Metrics

To evaluate the performance of regression trees, we will explore various ensemble methods discussed earlier and compare them to a single pruned regression tree, which will serve as our baseline model. For each method, the mean squared error (MSE) will be calculated as the primary error metric. Since classification and regression trees are valued for their interpretability and ease of visualization, we will include plots and visualizations to support the interpretation of the trees and their key features. All data processing and modeling will be conducted using Python and relevant libraries.

4. Results

4.1 Regression Tree with Pruning

Initially, a regression tree was built without imposing any constraints on tree depth or complexity. This resulted in a highly complex tree with a depth of 29, utilizing all 297 features

(see Figure 1). The unpruned tree achieved a test mean squared error (MSE) of 1,518,741,917 and a corresponding root mean squared error (RMSE) of 38,971. Considering that the mean and standard deviation of the sales price is 186,406 and 79,502, respectively, an average prediction error of 38,971 should be considered reasonable. While the model achieved a reasonable RMSE, the excessive complexity of the tree raised concerns about overfitting and interpretability.

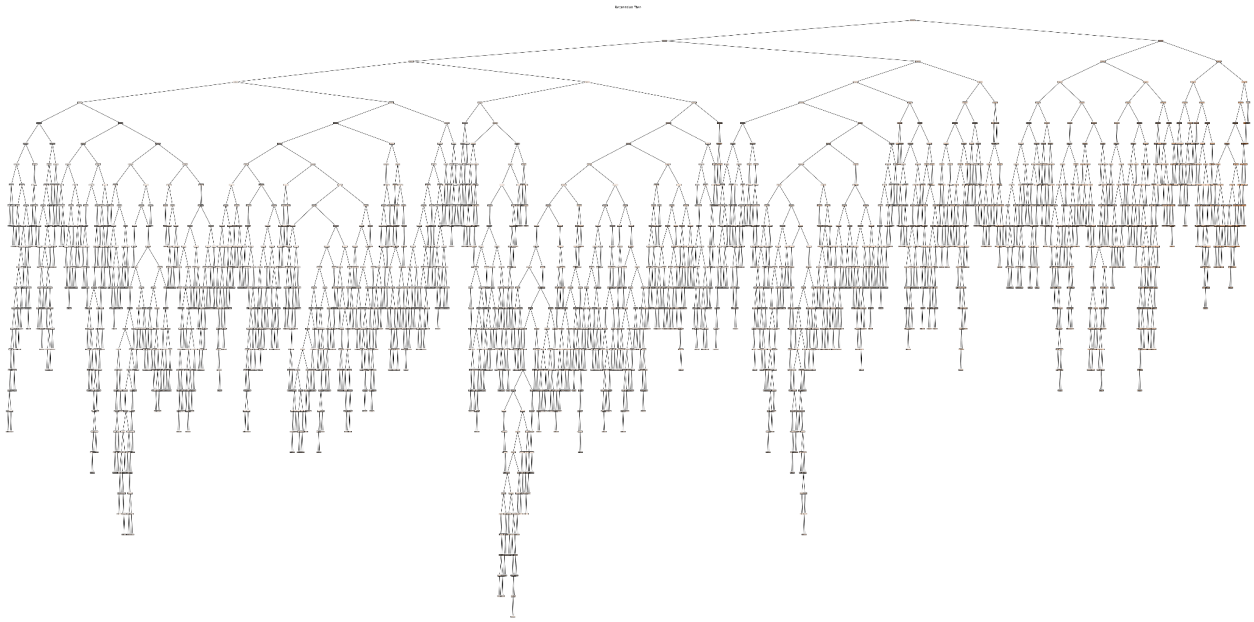

Figure 1. Visualization of regression tree with no additional

To address the overfitting issue, cost-complexity pruning was applied to the regression tree. Using the tree's cost-complexity pruning path and cross-validation with 5 folds, an optimal pruning parameter (ccp_alpha) was identified. The pruned tree had a depth of 8 (see Figure 2.), significantly reducing its complexity while maintaining predictive performance. The pruned model achieved a test MSE of 1,366,036,500 and an RMSE of 36,960, showing improved generalization compared to the unpruned tree.

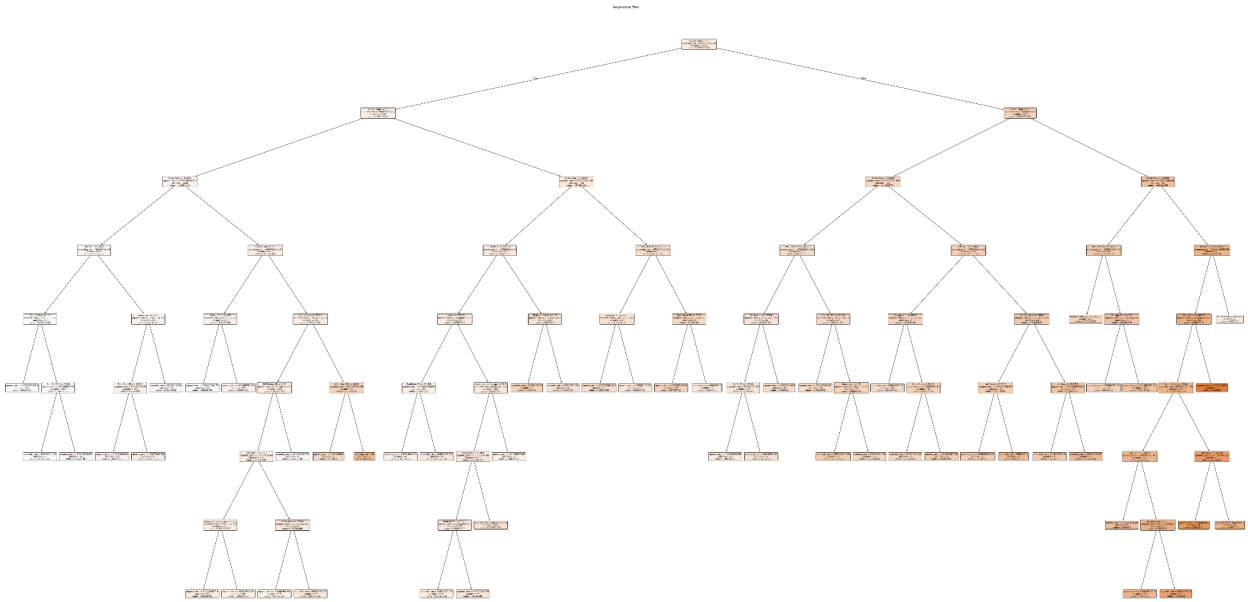

Figure 2. Visualization of regression tree with pruning

4.2 Bagging

This resulted in an MSE of 745683469.9055943 which means that the average deviation from the true sales price from our model is about $27307. Which is an improvement from the pruned tree.

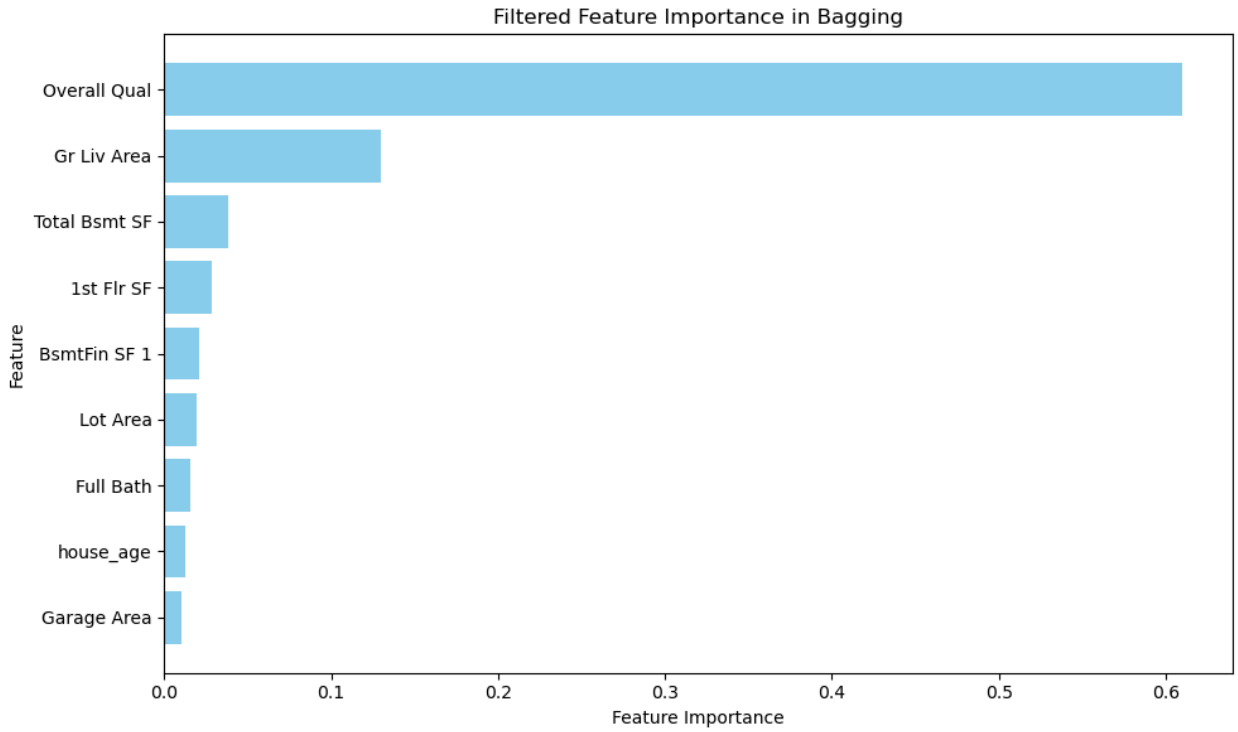

Figure 3: Feature importance plot - Bagging

4.3 Random forest

We can see that in our case this resulted in a MSE of 656592272.2457263 which means that we were off with about $25624 from the true value of the sales price. A small improvement from bagging. We also notice that the plot of feature importance now contains a lot more features than the previous.

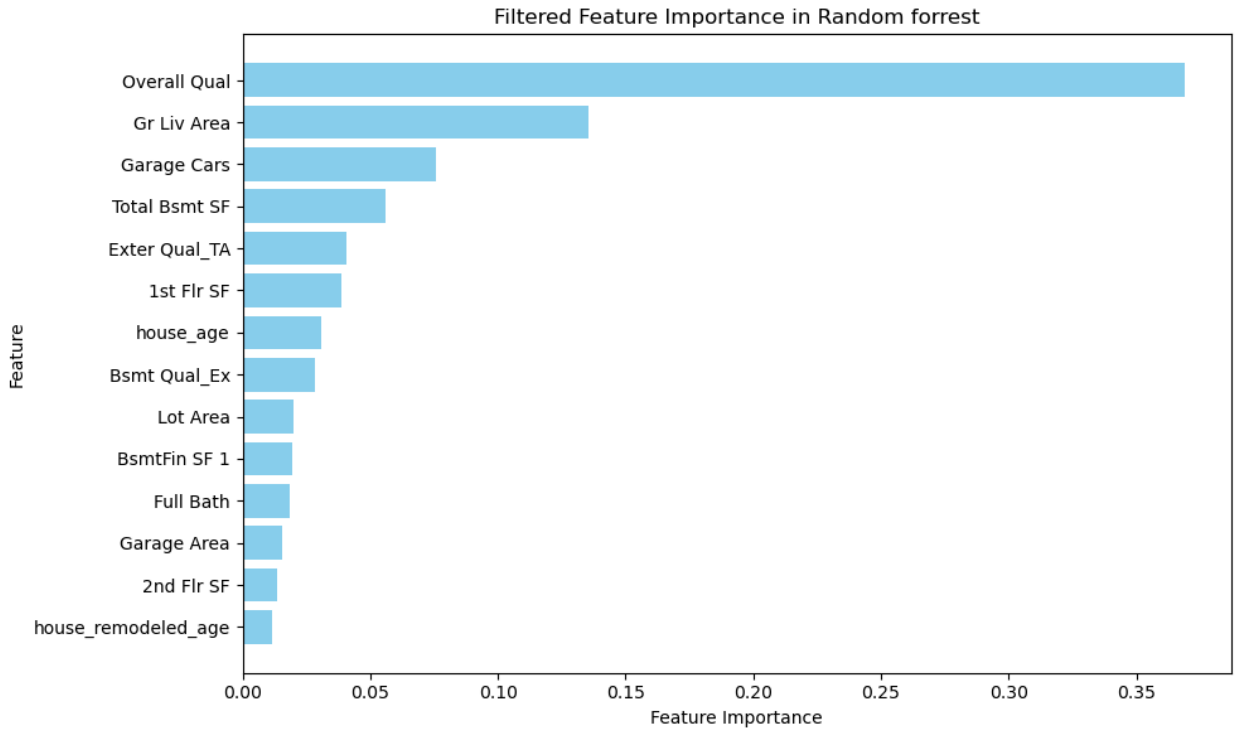

Figure 4: Feature importance plot - Random Forest

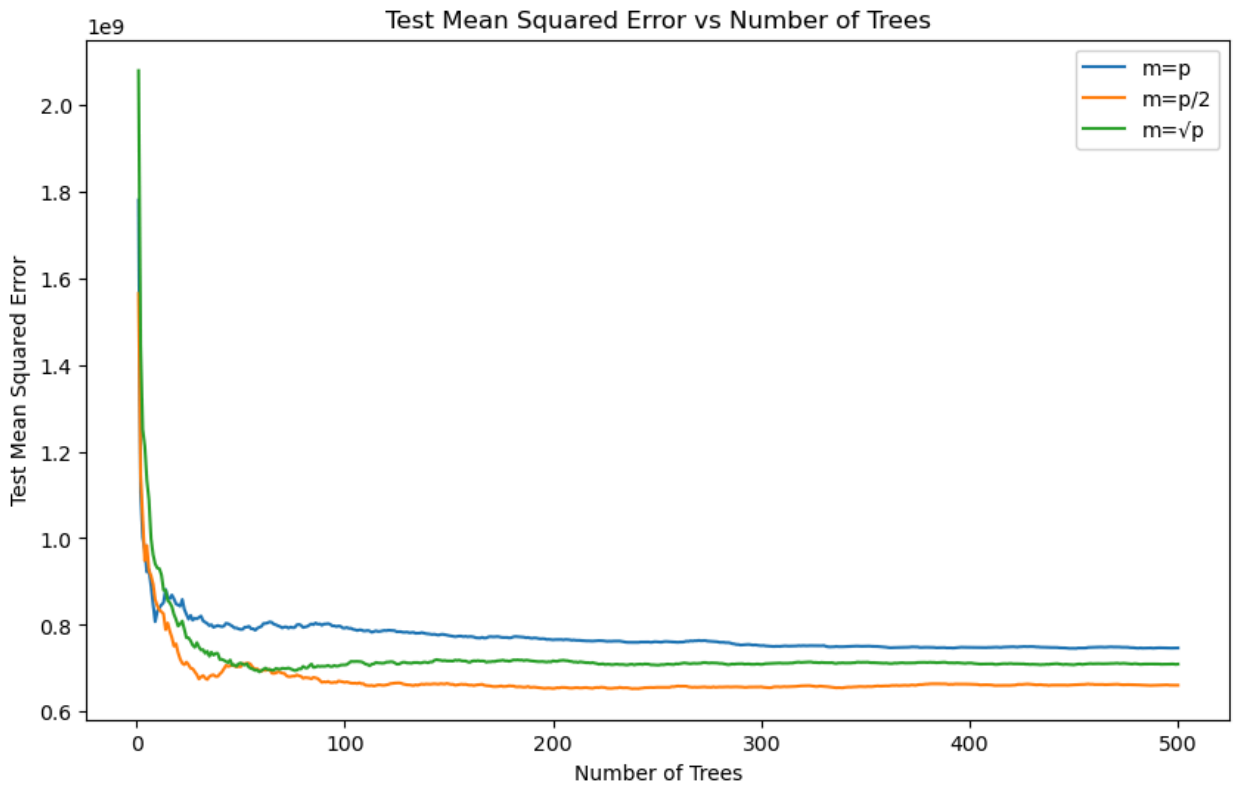

Figure 5: Random Forest; Number of features vs MSE

4.4 Boosting

4.4.1 Gradient boosting

Next we consider a gradient boosted tree. These trees are built sequentially and slowly, where each tree works on the struggles from the previous tree. This resulted in a squared MSE of 25068, which is a slight improvement from the random forest.

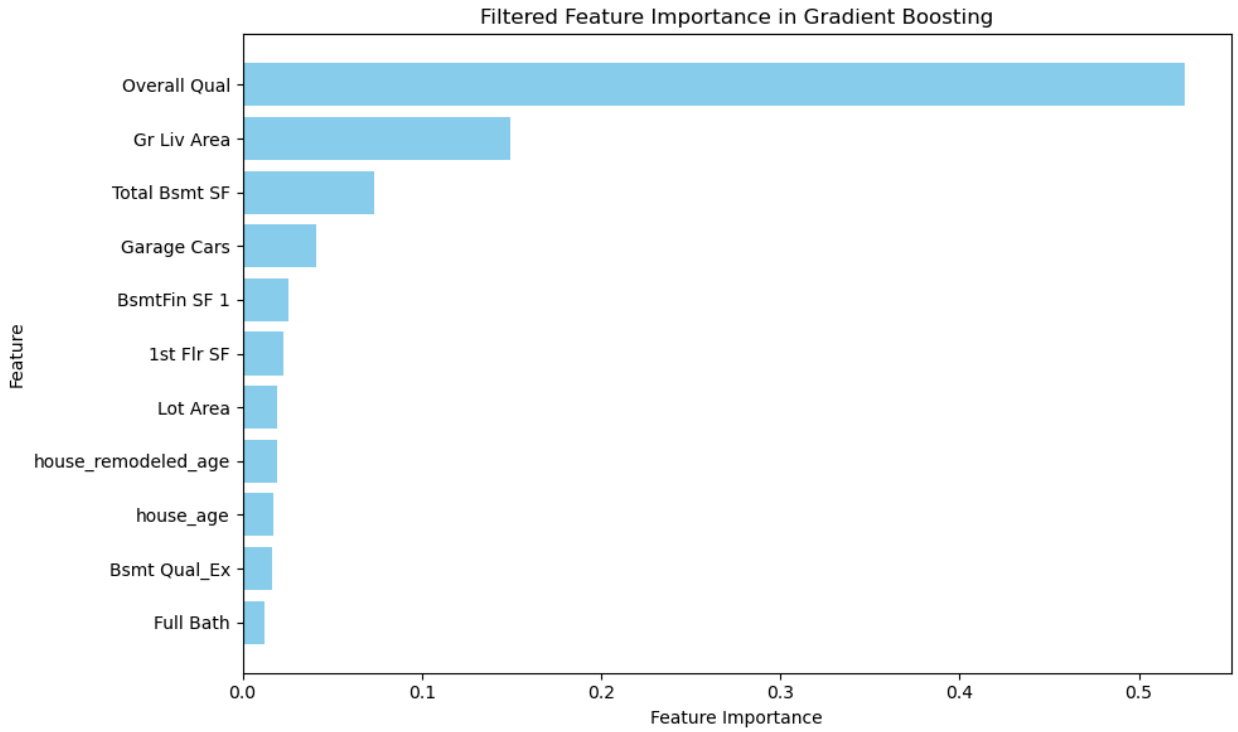

Figure 6: Feature importance plot - Gradient Boosting

4.4.2 XG boost

Finally we used XG boost which is known to be an improvement over regular gradient boosting. For this we got an MSE of 455916490.4695 which translates to an average deviation of 23268$ from the true salesprice. Clearly, this method performed the best out of all tested.

5. Conclusion & Discussion

Table 1. Performance of all tree based models using MSE and RMSE as performance metrics.

Model	**MSE**	**Root MSE**

Pruned tree	1,366,036,500	36,960
Bagged tree	745,683,470	27,307
Random forest	656,592,272	25,624
Gradient boosted tree	628,404,624	25,068
XG Boost	455,916,490	23,268

From Table 1. we can clearly see a decrease in error from our first original pruned tree. The biggest decrease happens when we go from the pruned tree to the first , more basic, ensemble method, namely *bagging.* Thereafter, the decreases from different methods aren't as steep. A reason for this might be, as stated in the introduction, that trees are sensitive to changes in the input data. Bagging, which essentially means drawing *new* data to create an ensemble of trees, seems to mitigate this problem by making the final aggregate predictions more robust and generalisable.

As we saw in the feature important plots, *Overall Quality* was always a feature that proved to be the most important. Using random forest to limit the effect of strong features and reduce the correlations did result in our feature importance plot including more variables but still had *Overall quality* as the most important. However, all in all it did reduce the test error when compared to regular bagging.

The lowest MSE we managed to achieve came when using *boosting,* specifically XG Boost. Our data contained a lot of features, especially after encoding many of them into dummy variables, making the model increase in complexity. XG boost can handle this very well since it makes use of regularization to prevent overfitting whilst sequentially building the trees to correct errors from previous trees. Another reason for it doing well can also be that there were subtle patterns in the data that were easily detected when slowly growing the trees sequentially.

Since we didn’t use any other model to compare our results too, we can still say that the methods that we introduced in the introduction were able to increase the prediction accuracy of regression

trees. The graph of our pruned tree as well as the feature important plots also provides easy to understand visualizations making trees a viable choice when you want interpretable results and not to make any assumptions about the data or parameters.

6. Reference

De Cock, D. (2011). Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. Journal of Statistics Education, vol. 19, no. 3, https://jse.amstat.org/v19n3/decock.pdf

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System, in B. Krishnapuram (eds), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York: Association for Computing Machinery, pp. 784-794, https://doi.org/10.1145/2939672.2939785

Elith, J., Leathwick, J. R. & Hastie, T. (2008). A Working Guide to Boosted Regression Trees, Journal of Animal Ecology, vol. 77, nr. 4, s 802-813

James, G., Witten, D., Hastie, T., Tibshirani, R. & Taylor, J. (2023). An Introduction to Statistical Learning: With Applications in Python. Cham: Springer.

Kuhn, L., Page, K., Ward, J. & Worrall-Carter, L. (2014). The Process and Utility of Classification and Regression Tree Methodology in Nursing Research, J Adv Nurs, vol. 70, nr. 6, s 1276-86

Loh, W.-Y. (2014). Fifty Years of Classification and Regression Trees, International Statistical Review, vol. 82, nr. 3, s 329-348

Palermo, M. (2020). Ames Iowa Housing Data, https://www.kaggle.com/datasets/marcopale/housing/data [Accessed 4 December 2024]

Simple Technique for Ecological Data Analysis, Ecology, vol. 81, nr. 11, s 3178-3192

Sutton, C. D. (2005). 11 - Classification and Regression Trees, Bagging, and Boosting. in: Rao, C. R., Wegman, E. J. & Solka, J. L. (eds.) Handbook of Statistics. Elsevier pp 303-329.