

Finding the Right Price:

COMPARING MACHINE LEARNING MODELS to PREDICT HOUSING PrICES in Ames, Iowa

Name: David Sarai

Student#: 501077059

CIND820

Supervisor: Professor Zekiye Erdem

Date: April 4, 2022

Table of Contents

[Abstract 3](#_Toc100524118)

[Literature Review 3](#_Toc100524119)

[Summary 13](#_Toc100524120)

[Project Methodology 14](#_Toc100524121)

[Description of Data Set and Prior Studies 15](#_Toc100524122)

[Data Preparation & Exploratory Data Analysis 17](#_Toc100524123)

[Data Preparation I 17](#_Toc100524124)

[Exploratory Data Analysis 19](#_Toc100524125)

[Data Preparation II 39](#_Toc100524126)

[Modelling 49](#_Toc100524127)

[Multiple Linear Regression 50](#_Toc100524128)

[Lasso Regression CV 50](#_Toc100524129)

[The Decision Tree Regressor 52](#_Toc100524130)

[Random Forest Regressor 55](#_Toc100524131)

[Gradient Boosting (XGBoost) 59](#_Toc100524132)

[Artificial Neural Network - Perceptron (Keras) 62](#_Toc100524133)

[Modelling Summary 67](#_Toc100524134)

[Conclusions 69](#_Toc100524135)

[Dataset Features 75](#_Toc100524136)

[Dependent Variable 75](#_Toc100524137)

[Discrete Variables 75](#_Toc100524138)

[Continuous Variables 77](#_Toc100524139)

[Ordinal Variables 79](#_Toc100524140)

[Data Appendix 86](#_Toc100524141)

[Effectiveness Results 86](#_Toc100524142)

[Results with No Standardization of Test Data 89](#_Toc100524143)

[Efficiency Results 90](#_Toc100524144)

[Hardware Comparison 90](#_Toc100524145)

[References 91](#_Toc100524146)

# Abstract

Alongside food and water, housing is a basic human necessity. It is then a concern that housing is increasingly becoming a scarce commodity.

Canada is experiencing an incredibly tight property market with demand and, consequently, prices soaring. While select large urban real estate markets such as Vancouver and Toronto had already been experiencing significant price pressure, elevated prices are increasingly becoming the norm across much of the country. According to the CREA(2022), year-over-year home prices across the country reached a record high of 26.6% in December 2021 and this type of upward price movement is expected to continue.

In this context, being able to accurately assess and predict real estate prices is of paramount importance. This is certainly true for prospective home buyers who need to make every penny count. Such insight is also in high demand from investors and developers who are under intense pressure to find value. From a very different perspective, public agencies and policymakers at all levels of government have a pressing need to be able to predict housing prices - especially in the context of the current housing shortage.

The main research question that this project seeks to answer is: ***Which current analytical methods and machine learning techniques perform best in predicting housing prices?*** A secondary question is ***whether more novel machine learning methods are a significant improvement on traditional regression methods and if so in what ways?***

To explore this question, I will use the **Ames Housing dataset**. This feature rich dataset documents housing prices in Ames Iowa from 2006 to 2010. It was assembled as a modern alternative to the well-known Boston Housing dataset that had long served as benchmark to assess linear regression models (De Cock, 2011).

The themes that this project deals with are **regression/classification**. The methods that I propose to compare include: multiple linear regression (baseline), Lasso regression, regression tree, random forest, gradient boosting and artificial neural network models. Prior to model comparison, a process of feature selection will likely be necessary using techniques such Lasso regression.

The main tools that will be used for the project will be Python along with various libraries such as pandas, sci-kit learn, XGBoost, and Keras.[[1]](#footnote-1)

# Literature Review

This section looks at the development of analytical methods of property pricing. I briefly review the origins and development of linear regression in relation to hedonic pricing and specifically predicting property prices. I then discuss the development of machine learning methods to property price prediction and compare these to the econometric methods of hedonic pricing. Finally, I discuss four methods of machine learning (Artificial Neural Networks, Regression Trees, Random Forest, and Gradient Boosting) that have been applied to property price prediction

Using methods of linear regression to predict property prices has a long history. The ordinary least squares method had been discovered in the early 1800s and by the middle of that century it had begun to be widely applied in the emerging social sciences. However, it was in the 1930s that this method began to be applied to the problem of property valuation - specifically farm prices (Colwell & Dilmore, 1999; Sirmans et al., 2005). Multiple regression proved to be an ideally suited technique for what would become known as the hedonic method of valuation.

One of the key problems with pricing real estate and housing is that each unit of property is most often very different from other units(Haider, 2016;Haider, 1999). Most goods in a modern economy are more or less standardized in form and hence can be priced according to supply and demand as a commodity. However, the heterogeneity of property makes it more difficult to price property in such a straightforward manner. Hedonic pricing tackles this problem by disaggregating a complex good like a piece of property into the demand for the individual components that make up that good. The price of a house then can be understood as the sum of a variety of different features such as square-footage, the number of bedrooms, or the size of the overall lot.

Though already in practical use, academic work on hedonic pricing of housing in economics really began to explode in the 1970s (Sirmans et al., 2005). There were two key reasons for this. First, economists developed and extended theoretical frameworks which would explain and underpin the practical application of hedonic regression. In particular, Lancaster provided a theoretical justification grounded in microeconomics on the utility generating characteristic of housing while Rosen worked out an understanding of price determination (Sirmans et al., 2005). The second factor that encouraged academic research on hedonic pricing of housing was the development and increasing accessibility of computer hardware. Even with computer technology in the early 1970s, a single regression analysis conducted on a massive mainframe computer that involved tedious card punching would take up to twenty-four hours(Ramcharan, 2006). As such, the rapid development and adoption of computer technology in subsequent decades would spur the growth of increasingly sophisticated econometric house pricing models. Besides incorporating elements such as market segmentation and spatial analysis, econometric models provided increasingly refined explanations that elucidate various aspects of the economic operation of the housing market (Goodman & Thibodeau, 2003; Haider, 1999).

Of course, the advance of computing technology also spurred the development of computer science in general and, in particular, artificial intelligence and machine learning. Beginning in the 1990s and accelerating thereafter, a growing number of studies sought to apply a variety of new approaches that originate in computer science and engineering to the problem of pricing real estate. As a result, there is now a sizeable and diverse literature on machine learning approaches to property price prediction.

When comparing econometric approaches with machine learning approaches to property pricing, four aspects standout. First, the starting point for machine learning approaches is generally the hedonic regression model. This basic model is then adapted to a variety of different algorithms whether it be mapping it to a neural network or applying it in the form of recursive partitioning via a regression tree. Machine learning models still retain the basic underlying measures such as mean squared error which then provide key points of comparison with other regression models. Second, it is clear that econometric modelling places a high priority on grounding models in formal mathematical, statistical and economic theory (i.e. theory-laden modelling)( (Boelaert & Ollion, 2018; Tchuente & Nyawa, 2021). In contrast, though based on mathematical principles, machine learning models tend to be highly practically oriented. Third, and most significant in the current context, machine learning models have proven to produce a high-level of predictive value when applied to the problem of housing prices. While not necessarily true in every instance, many different researchers have found that in most cases machine learning models outperform multiple linear regression models in this area (Mayer et al., 2019; Tchuente & Nyawa, 2021; Valier, 2020). Fourth, the predictive power of machine learning approaches comes at the expense of explanatory power. Boelaret and Ollion (2018, p.10) point out that quantitative methods in social science have a number of typical modes of explanation such as : “statistical significance of an effect (controlling for other predictors, gender has a significant effect on wages, at the 5% level’), the sign of this effect (‘women earn less than similar men’), its magnitude (‘women earn a% less than similar men’), or a confidence interval for this magnitude (‘the gender gap is between b% and c% with probability 95%’).” In contrast, the complexity of many machine learning approaches along with their non-transparent structure often makes it difficult to trace out models’ decision-making processes and renders modes of interpretability and explanation ineffectual (Hoepner et al., 2020). As such, it is important to keep in mind that machine learning is, in important respects, different from econometric analysis and therefore not simply a replacement for such quantitative analysis.

Nevertheless, the sheer predictive power of machine learning approaches has made them irresistible for highly practical problems such as prediction of housing prices. This is especially true in the context of dramatic increase in the quantity, velocity and complexity of information that has been unleashed by the internet and social media. Further, the successful application of machine learning techniques in this area has spurred increasing interest and innovation.

A review of the literature on machine learning highlights four different methods which researchers have suggested improve upon prediction of property prices via traditional regression. These are: Artificial Neural Networks (ANN), Decision Trees, Random Forest, and Gradient Boosting. In fact, many recent articles are structured around comparing the performance of several of these machine learning techniques along with the standard multiple regression model in terms of measures such as mean square error (MSE), root mean square error(RSME), mean absolute error (MAE).

#### Artificial Neural Networks

Inspired by a renaissance of research in the area of neural networks in the mid-1980s, ANN was one of the first advanced machine learning techniques to be applied to the problem of property pricing(Valier, 2020). Modelled on the human brain neuron, a neural network consist of several layers: an input layer(independent variables), 1 or more hidden layers(computation nodes with various weightings), an output layer(final computation node that produces result) (Do & Grudnitski, 1992; Worzala et al., 1995). As opposed to arriving at the best-fit regression equation via the least-squares method, the neural network arrives at the weighting of various features iteratively. The data in the input layer is fed forward through the weightings to various computation nodes that comprise the hidden layer. These weightings are then adjusted in relation to the actual values in the training data via a function like gradient descent before again feeding the input data through the hidden layer. The optimal values are then arrived at through successive iterations of forward-feeding and backward propagation.

A number of studies that have found that ANN results in superior performance to traditional hedonic regression when predicting the price of houses using the same dataset (Do & Grudnitski, 1992;McCluskey et al., 2013; Tay & Ho, 1992). For example, when looking at a Turkish housing dataset containing 5741 observations and 41 features, Selim(2009) found that a neural network with two hidden layers significantly outperformed a semi-log hedonic regression model in terms of mean square error (MSE), root mean square error (RMSE), and mean absolute percentage error(MAPE). These findings accord with those of other researchers who suggest that the self-teaching quality embedded in the neural networks is better able to handle complex potentially non-linear relationships with potentially interrelated features than traditional hedonic regression (Limsombunc et al., 2004; Valier, 2020). Nevertheless, ANN also has a variety of short-comings. One of the most significant problems with ANN is the black box nature of the model (McCluskey et al., 2013). That is to say that the precise way in which the model arrives at predictions is not transparent. Therefore, while its predictive value may be high, the explanatory power of the model tends to be low. A number of researchers have also pointed out that depending on the dataset and the optimization of the neural network, in practice ANN’s improvement over traditional methods of linear regression are not necessarily significantly better than various advanced methods of multiple linear regression (McCluskey et al., 2013; Worzala et al., 1995). In particular, the trial/error optimization of ANNs are often tricky in practice and to be most effective an ANN needs large datasets. Mayer and his co-authors (2019) also point out that while the predictive accuracy may be high, ANN models can also be highly volatile which can be very problematic in practical contexts (e.g. property appraisal).

In spite of these drawbacks, there has been much recent interest in ANN models for property price prediction. Specifically, many new approaches which combine elements such as computer vision and image processing are built out from ANN frameworks (Poursaeed et al., 2018; Wang et al., 2021).

#### Regression Trees

Regression trees are another machine learning method that has been applied to the problem of predicting housing prices. Regression trees are very similar to decision tree classifiers but are applied to the problem of predicting a quantitative dependent variable as opposed to a categorical variable (Peng, 2021). Both types of trees are built via a process of recursive partitioning a dataset by its features. Zurada, Levitan, and Guan (2011) point out that regression trees are then essentially “ordinary decision trees with linear regression models at the leaves that predict the value of observations that reach the leaf”. While decision trees use an impurity measure such as a GINI or entropy measure to select features that maximize information gain, regression trees instead base their optimal splits upon minimizing expected error (i.e. minimizing standard deviation) until the result of such splits would no longer be statistically significant. The regression tree’s leaves are then pruned to prevent overfitting the training data (alternatively the maximum number of tree levels may be specified).

Several studies have shown that regression trees can be successfully applied to the problem of housing prediction(Fan et al., 2006; Özsoy & Şahin, 2009). In particular, Zurada(2011) and his co-authors show that in a variety of scenarios M5P type decision trees significantly outperform not only traditional linear regression but also a variety of ANN methods and, to a more limited extent, Support Vector Machine Regression. Like ANN, regression trees can handle data that does not necessarily meet the basic requirements for linear regression and can readily adapt to non-linear data (Özsoy & Şahin, 2009). Researchers have also pointed out that regression trees offer significant explanatory value given that one can identify the key variables that partition the data by examining the resulting visualization that is produced by the model.

However, regression trees also have some significant weaknesses. Fan, Ong and Koh (2006) point out that, in selecting the particular feature that is most significant at any node point, a regression tree will necessarily tend to overlook the not insignificant effect of other independent variables at a given node point. Due to this issue, in practice, regression trees can produce unstable results as relatively small variations in training data can produce quite different results. As well, regression trees tend to overfit training data which necessitates putting limits on the tree structure via pruning. It is not surprising then that, in a recent study of housing prices in Africa, the regression tree approach was still the poorest performer next to multiple linear regression when compared with robust regression techniques (Lasso and Ridge), random forest, and gradient boosting (Embaye et al., 2021). Indeed, in this body of literature focused on prediction of property prices, regression trees tend to be overshadowed by random forests.

#### Random Forest

Random forest is an ensemble approach that builds upon the regression tree approach. As its name suggests, a random forest model relies, not on the results of a single regression tree, but the aggregate result of many trees (Peng, 2021). By averaging the result of this collection of regression trees, the forest can produce more stable results that avoid overfitting the data. There are two additional key aspects that boost the predictive performance of this approach – bagging and randomizing features fed into the splitting algorithm. Both work to decorrelate the individual results of the various regression trees. Bagging works by using random sampling with replacement (bootstrapping) to feed the various trees that make up the forest with many possible variations of the data for the purposes of training the models. To further differentiate the overall forest, trees are fed a random number of features when making splits rather than assessing all possible features. Together these two aspects differentiate both the data and features that each tree assesses. As result, the differing trees tend not to reproduce the same prediction errors which produces more accurate overall prediction when individual outcomes are averaged.

As a result of improving and building upon regression trees, random forest approaches have been found to be among the better performing machine learning methods applied to the problem of property/housing price prediction. As such, in the literature they are far more commonly used than simple regression trees. For example, the study by Ceh et al.(2018) of apartment prices in Ljubljana Slovenia (7407 observations, 36 features) shows that the random forest approach produced demonstrably better results in terms of MAPE than multiple linear regression. A more recent study by Hong, Choi and Kim(2020) using a larger data set from Gangham, South Korea (16,601, 26 features, from 2006-2017) found that random forest performance was demonstrably superior in a variety of respects to an ordinary least-squares model of the data. Likewise, the findings of Peng’s(2021) study of Japanese housing prices show that random forest performs better than multiple regression in terms of both RMSE and MAPE. Perhaps more interesting, are several recent studies that suggest random forest performs well when compared to other contemporary machine learning methods. Ho, Tang and Wong’s (2020) study of over 30,000 housing transactions in Hong Kong compares the prediction performance of SVM Regression, random forest and gradient boosting. They found that random forest and gradient boosting both performed almost equally in terms of MSE, RMSE and MAPE and both significantly outperformed SVM-Regression. Other comparative studies focusing respectively on several African countries and China also suggested that random forest, though not the top model, is a strong predictor (Embaye et al., 2021; Xu & Li, 2020). In addition, an extensive literature review by Malaysian researchers of diverse comparative studies of machine learning approaches to property prediction found that random forest consistently was identified as a top performing model (Ja’afa et al., 2021). Nevertheless, random forest does sacrifice the explanatory performance of the regression tree model as utilizing the aggregate performance of regression trees then obscures the role that individual features play in determining the outcome of the model. As well, Mayer et al.(2019) point out that, in practice, random forest models often have high biases and difficulty detecting market trends and dealing with historical data.

#### Gradient Boosting

Gradient boosting is another ensemble machine learning approach that has provided high performance in terms of prediction of housing/property prices. In contrast with the randomized approach of random forest, gradient boosting works through sequential improvement (Mayer et al., 2019). The key idea is to generate a robust aggregate model through continuous improvement over a sequence of weaker learning models. Specifically, gradient boosting approaches most often use a succession of relatively shallow decision trees with a limited number of levels, nodes, or leaves. Initially, a single simple regression tree is generated from the data. When creating the next simple tree, the emphasis is placed on the aspects of the data set that the first tree had the most difficulty with. By means of a gradient descent function the weightings of the data are adjusted to focus on the residuals of each previous tree in the succession. In this manner, the mean squared error is progressively lowered until further improvement is exhausted. At this point, the predictions of all the simple trees are averaged to provide an aggregate result. The idea is that each simple tree concentrates on a particular aspect of the data and all the areas where it is weak are explained by other trees in the succession (i.e. each tree concentrates on a portion of the overall complexity embedded in the data). The result is a very strong predictive model which usually provides greater accuracy than random forest (Mayer, 2019). However, like random forests, gradient boosting also is something of a black box as it is difficult to explain and interpret the composite elements that make up the model’s output.

In the literature, gradient boosting approaches are consistently among, if not the best, performers in terms of property price prediction. As this approach is newer, there are fewer examples of it in the literature. Nevertheless, among the articles reviewed here, gradient boosting scores as the best predictive model in every study in which it is included. It is true that gradient boosting’s advantage in Ho, Tang and Wong’s(2020) study of Hong Kong housing prices is extremely slight when compared with the performance of random forest in terms of RMSE. However, in the case of Embaye, Zereysus and Chen’s(2021) study of the rental value of homes in Tanzania, Uganda and Malawi, gradient boosting is the strongest model in all three years of the study in terms of having the lowest mean squared error. Likewise, Xu and Li(2020) found that XGBoost, a gradient boosting package, scores the lowest RSME in terms of both the test and training sets of the base models in their study of the “second-hand” housing market in major Chinese cities. Unfortunately, other recent comparative studies of machine learning price prediction leave gradient boosting out(Valier, 2020).

## Summary

The preceding review suggests there are some evident trends in terms of the performance of machine learning models such as the strong performance of gradient boosting and the relatively weak performance of individual regression trees. Nevertheless, in the current literature there is not simply a definitively best machine learning approach to property price prediction (Ho et al., 2020; Valier, 2020). In part, this may be due to the fact that, though there is a voluminous literature on machine learning property price prediction, there are very few studies that are easily comparable due to use of different data sets or choice of model for comparison. There also seems to be very few attempts in the literature to reproduce the findings of past studies. It is also likely that the sheer complexity of machine learning also prevents straight-forward conclusive comparison. With so many different techniques for data preparation, software platforms, and implementations of various algorithms, finding a sound basis for comparison is simply difficult. The most significant issue might be simply the breadth of research and pace of innovation in machine learning. Indeed, there are a myriad of recent studies on ML property price prediction which combine approaches such as ANN with new techniques drawn from other areas of data science and computer engineering such as sentiment analysis and computer vision (Hausler et al., 2018; Poursaeed et al., 2018; Wang et al., 2021). As a result, both the model and the data analyzed are becoming increasingly complex. Needless to say, this also makes any kind of strict comparison difficult. As such, one might agree with Ho and his co-authors (2020) who conclude that at present the focus in ML property price prediction should be eminently practical – find what is the best approach in the given circumstances (e.g. size/complexity of data, available computing resources, available expertise, purpose of project).

# Project Methodology

Figure Project Method

# Description of Data Set and Prior Studies

This section provides a summary of the Ames housing dataset and also reviews the ML literature specific to this data.

The Ames housing dataset contains property assessment data for homes sold in the City of Ames, a small city in Iowa of 66,427, between 2006 and 2010 (De Cock, 2011). It consists of 2930 records with 81 features (see tables) in addition to the dependent variable sale price which we are trying to predict. Of these features, two are identifiers – a unique data identifier as well as the City of Ames’ property identification number. The remaining features breakdown as follows: 23 nominal, 23 ordinal, 19 continuous, 9 discrete, 5 date.

The Ames housing dataset is very well-known and popular. There are numerous web sites that discuss the dataset or use it to illustrate various data science concepts. As well, a portion of the Ames housing dataset features in a continuously open revolving Kaggle competition that thousands of teams have entered.

Surprisingly, there is very little published academic literature which primarily focuses on analysis of the Ames housing data.[[2]](#footnote-2) My search turned up four papers. The first of these by Simlai(2017) was an econometric analysis that uses the Ames data to model spatial idiosyncratic risk. While certainly a complex analysis, it is primarily an explanatory model that does not directly related to machine learning price prediction.

Two other papers use the Ames data to create regression models. In the first of these, Abdulhafedh(2022) performs a basic step-by-step project that progresses from data cleaning to exploratory analysis before creating a basic multiple regression model using R. He uses several different techniques including paired association, correlation, and random forest to reduce the number of features to 12. The resulting model was able to generate a R2 value of 0.9283 and a RSME of 0.12792 while cross-validation using both K-Fold and Leave-One-Out produced a Mean Squared Error of 0.12. Unfortunately, there is no comparison with alternative models. The second paper by Xin and Khalid (2018) is a very short paper that uses the Ames dataset to compare the effectiveness of ridge and Lasso regression. The paper contains no real exploratory data analysis. They find that the Lasso regression provides the better performance after selecting 19 variables with a RSME of 0.1225798 compared to ridge regression RSME of 0.1333643. Though this paper is fairly thin, it does contain some important insight into feature selection using Lasso which seems pertinent to my project.

The final paper by Singh, Sharma and Dubey (2020) is somewhat thin and uneven but it is of particular significance as it details a very similar project to the one that I am undertaking. Using R and associated libraries, the authors seek to compare the performance of multiple linear regression, random forest and gradient boosting. For feature selection, they propose using Lasso regression; however, oddly very little detail is given about how this Lasso regression is implemented in practice and it is not altogether clear what features are selected. In terms of model performance, gradient boosting was found to be the best performing model with an RMSE of 0.1128 while random forest scored 0.1499 and linear regression finished a distant third with a RSME of 0.5188. There is also very little mentioned about the process of cross-validation. In spite of the shortcomings of Singh, Sharma and Dubey’s paper, my project charts much the same course and therefore attempts to reproduce their results using Python and associated libraries while also adding a neural network and regression tree approach to the comparison.

# Data Preparation & Exploratory Data Analysis

## Data Preparation I

After developing a basic understanding of the dataset and reviewing the literature on machine learning approaches to housing price prediction, the next stage was the initial data preparation before moving on to exploratory data analysis followed by a second round of data preparation.

The initial data preparation consisted primarily of data cleaning. This task was relatively straightforward as the Ames dataset was largely in a technically correct form with consistent labels and datatypes. After ensuring that the dataset was read correctly and examining the basic summary statistics, a few other simple tasks were required. These included performing a check on the dimensions of the dataset and confirming that there were no duplicate records.

The largest portion of the data cleaning process involved identifying, examining, and resolving not available (Na) values in the dataset. Many of the features which might be presumed important such as Total Above Ground Square Footage (GrLivArea) or Lot Area (LotArea) had no Na values. In contrast, some features including PoolQuality and Fence had Na values for a majority of the records. In most cases, Na values simply indicated an absence of that feature for the particular property. This often could be confirmed by a zero value in an associated column. For example, houses that had an Na for Pool Quality in all cases also had zero for PoolSquareFootage. Likewise, properties that had Na values for BasementCondition had zero for TotalBasementSquareFootage. With such Na values, it was appropriate to simply replace the Na value with None.

There were, however, Na values for a few features that seemed more problematic and required further investigation. One example was a single home that had a Na value for Electrical. It was highly unlikely that a modern home in the United States would lack an electrical system. Given that the home was built in 2006 it is almost certain that the home would have a standard breaker box as no home in the dataset built after 1965 had any other type of electrical system.

A more significant mystery was the fairly large number of homes (490) that had Na values for Lot Frontage indicating the square footage of the lot that was connected to a street. This seemed strange as one would assume that most homes would have some portion of their property that connected them to a street. Of course, it is possible that these Na values represented apartments, condominiums, or other multistory dwellings. However, inspecting these records revealed that the majority of homes with Na values for Lot Frontage were, in fact, single family homes. Fortunately, it was possible to obtain more information about these homes using web accessible property records of the Ames City Assessor. An examination of a sample of ariel photographs of lots revealed that almost all of these homes had some portion of lot frontage which was indicated in square footage on the pictures.

Map

Description automatically generated

It appears then that the Na values for Lot Frontage represented missing data. Given that it was not practical to calculate the frontage for all these homes, K-nearest neighbors (scikit-learn’s KNN imputer) was used to impute a square footage value for frontage.

The next step in the data cleaning was to address some of the basic coding for categorical features. For example, it was necessary for many ordinal features to replace their levels (e.g. Poor, Good, Excellent) with numeric values. There were also a couple of cases of nominal variables where numeric identifiers had to be changed to non-numeric values. In addition, I simplified a number of ordinal variables by reducing the total number of levels on the assumption that it would improve model performance as the very high and very low levels contain very few examples. However, after some modelling, I decided against using these simplified features. Finally, the two identifier columns were removed as they were redundant.

## Exploratory Data Analysis

After cleaning up and ensuring the consistency of the dataset, it is now possible to move on to the exploratory data analysis. We begin by investigating the relationship between our target variable, Sale Price, and a number of other numeric features which we assume to be highly determinative of the cost of a home. In this regard, the most obvious feature is the size of the house in terms of the square footage of the main living area.

Chart, scatter chart

Description automatically generated

A scatter plot of Total Above Ground Living Area and Sale Price shows that there is a fairly linear relationship between the variables. Of course, it makes sense that larger houses sell for more than smaller houses. The bulk of the data is concentrated in a range between approximately 800 and 2000 square feet at a cost of between $90,000-$275,000. What really stands out on this chart is a small number of extreme outliers. There are three homes that have more than 4500 square feet of living area yet all three seem to be priced at below $200,000. In contrast, the only two other homes with over 4000 square feet of living area are priced well over $700,000 and these prices seem in line with the rest of the data. It will be important for us to remove the outliers.

The other feature that one might assume to be highly predictive of sale price would be the overall size of the property on which the home sits.

Chart, scatter chart

Description automatically generated

While the relationship between Lot Area and Sale Price is fairly linear, the line has a very steep and only slightly positive slope. The data is even more concentrated than it was for living area. Almost all of the data is concentrated approximately in the range of 5000-10,000 square feet with a very wide variation in price. Upon reflection, this makes sense as the size of a lot in most urban areas in the United States is highly standardized. As well, unlike property used for commercial or industrial purposes, the square footage of the actual land area the home sits on does not provide increased income with increased size. Again, there are a few extreme outliers along with a larger number of more moderate outliers.

Next, we look at Total Basement Square Footage and Sale Price.

Chart, scatter chart

Description automatically generated

With this scatter plot, we see a linear relationship although a slight upward curve is perceptible. The majority of the data is concentrated in a range between 500 and 1750 square feet and is priced in the range of $80,000-$300,000. Complicating this picture, though, is a small number of data points that form a perfect upward line at zero square feet representing homes that don’t have a basement. Like the previous scatter plots, a small number of extreme outliers are evident.

With the last scatter plot, I thought it would be interesting to combine Total Above Ground Square Footage and Total Basement Square footage and then chart the relationship with Sale Price.

Chart, scatter chart

Description automatically generated

The scatter plot reveals a very clear linear relationship between Total Square Footage and Sale Price. It appears to be a stronger relationship than produced by either of the individual features and Sale Price. Further, the problem of the number of homes with zero values for basement is no longer apparent. Of course, the problem of a few extreme outliers remains.

Moving on with the exploration we shift to look at some categorical variables. One thing that became very clear during the data cleaning process was that there were several categorical variables that only had positive values for a very small number of properties. As well, within many of the categorical features there are levels that contain only a very few values. All of these will likely not play a large role in the modelling. There are, however, a number of features which would be very interesting to potential home buyers that we will explore with a series of box plots.

The first of these shows the various styles of houses available in Ames and the Sale Price.

Chart, box and whisker chart

Description automatically generated

We can see that it is possible to purchase nearly any style of home for a price in a band of about $140,000 to $200,000 in Ames. The price band for one-story and two-story homes varies considerably starting under $50,0000 and going up to the mid-$300,000s. Interestingly, while 2.5 Finished and 2.5 Unfinished price bands are higher than that for 2Story homes, 2Story homes have a significantly higher ceiling. The remaining house styles have relatively tighter pricing bands.

Next, we look at the influence that neighborhoods in Ames have on house prices.

Chart, box and whisker chart

Description automatically generated

Immediately it is apparent that Stone Brook, Northridge Heights, and Northridge are the priciest neighborhoods with median home prices over $300,000. Green Hills is only slightly cheaper. Otherwise, homes can be purchased in most neighborhoods within our price band of $150,000 to $200,000. Aside from the upper-class areas, only the upper-middle class neighborhoods of Somerset, Timber, Veenker and College Crescent have median prices above the $200,000 mark. Cheaper neighborhoods are Briardale, Meadow Village, Brookside, Old Town, Edwards, and South West of Iowa State University. Unsurprisingly, the Iowa DOT and Railroad is the cheapest place to buy a house in Ames.

Chart, bar chart

Description automatically generated

The chart above gives us a slightly different perspective on neighborhoods in Ames. We can more clearly see the average price of homes in various neighborhoods and the frequency of homes in each neighborhood along with a median line for home prices across all of Ames.

In this study, we are really looking at home prices within a short four-year period of time -from 2006-2010. As such, it is a snapshot of a particular period rather than any kind of account of housing price trends over time. Nevertheless, there are a few interesting insights that we can glean from the features list that are listed in years. For example, it is interesting to note that the majority of houses sold in our snapshot were built after 1960 with a very large number built in the late 1990s and early 2000s following a dip in the early 1980s.

When we look at Sale Price with Year Built, it is as expected that newer homes generally sell for more than older homes. What is perhaps more interesting is the two price segments that appear in the data. The median price of homes built before 1986 is remarkably stable at around $140,000. However, the median price for homes built 1986 and after shoots up to approximately $220,000.

Chart, histogram

Description automatically generated

If we look at the average and median prices of homes sold between 2006 and 2008, it is very noticeable that there is a decline in the price of homes after 2007 despite a slight uptick in 2009.

Chart, line chart

Description automatically generated

This runs contrary to the expectation for nominal housing prices to rise. However, the decline does fits with the general deterioration in housing prices across the United States that is associated with the US financial crisis in 2008. We can see in this chart from the Federal Reserve how Ames housing prices during this period fits with the wider housing price trends in Iowa.

Chart, line chart

Description automatically generated

The third stage of the exploratory data analysis shifts to look at the frequency distributions of various features. We begin by examining the distribution of our target variable -Sale Price.

Chart, histogram

Description automatically generated

It is evident that the distribution for Sale Price is unimodal with a right-skewed distribution with a steep curve. This is what one might have anticipated given that Sale Price had a skew of 1.744 and kurtosis of 5.119. The probability plot confirms that the distribution diverges from a normal distribution.

Chart, line chart

Description automatically generated

This does present something of a problem as ideally a normal distribution is optimal for linear regression models. Given the distribution of Sale Price, a traditional solution might be to perform a log transformation which would shift the distribution to the right.

Chart, histogram, scatter chart

Description automatically generated

There are a few things that stand out when looking at the histograms for the numeric features in the dataset. First, we see that several of the features such as Lot Area, Total Basement Square Footage and Total Above Ground Living Area are also right-skewed. A log transformation should also be useful in these cases. Second, there are quite a few features that have a large number of zero values that tend to overshadow the distribution. These features are comparable or often directly related to categorical variables with a large number of None values (e.g. Pool Quality=None, Pool Area=None).

A picture containing table

Description automatically generated

Moving on with the exploratory data analysis, we examine several correlation matrices. Here we are interested in seeing which features are most closely correlated with Sale Price. As well, it is important to see if any of the independent variables are closely correlated with other independent variables and therefore signify problems of multicollinearity.

To begin with we have a huge reverse heat map that contains all continuous, discrete and ordinal variables.

Graphical user interface, chart

Description automatically generated

While a bit overwhelming, it is possible to distinguish several different points of interest. First, it is quite clear that there are a few areas where multicollinearity is a problem. Many of these are very straightforward as they simply represent different aspects of a particular element. It is not surprising that Pool Quality is very close related to Pool Area, that Garage Area is related to the garage’s car capacity, that Total Rooms Above Ground is related to Total Above Ground Square Footage. It also makes sense that the year a house is built is very closely related to the year the garage was built given that most homes are built with garages. The negative correlations also follow a similar pattern. If one has a largely unfinished basement, not only will they have very low finished basement square footage but also a low number of full bathrooms in the basement.

The next correlation matrix zooms in specifically to look at the variables that are most highly correlated with Sale Price.

Chart, bar chart

Description automatically generated

The top correlations with Sale Price are as we would expect: Overall Quality, Total Above Ground Square Footage, Exterior Quality, Kitchen Quality, Garage Cars/Area, Basement Square Footage, and Basement Quality. This confirms that people buy houses based on how appealing a home is and its size. The rest of the top correlations are other priorities for home buyers that are of slightly lower importance such as fireplaces, the number of bathrooms and the quality of the heating system.

Next is a correlation matrix strictly of numerical features. We have already discussed many of the variables when looking at the heat map. Here I have included the correlation coefficients.

A picture containing text

Description automatically generated

The last correlation matrix examines ordinal features using the Kendall Rank Method. The Kendall method is far more appropriate than the Pearson method as the values of these features simply represent a rank ordering. The matrix shows that there is not a lot of monotonicity between the features. The one exception is the rather strange link between Exterior Quality and Kitchen Quality. Why would the quality of the outside of a home be closely related to the home’s kitchen? The answer to this question likely has to do with remodeling and updating homes. Often a homeowner who has remodeled the exterior of his or her house also quite likely updated other parts of the house. This hypothesis is supported by the relatively high correlation between Year Remodel/Addition and both Exterior Quality and Kitchen Quality.

Chart

Description automatically generated

The exploratory data analysis ends with a combined pair plot and histograms of the variables that are most correlated with Sale Price.

A picture containing diagram

Description automatically generated

This chart shows that there is a lot of linearity in the data (with Sale Price) particularly for key variables like Total Above Ground Square Footage.

## Data Preparation II

Coming out of the data exploration, a brief second round of data preparation is needed before moving on to feature selection. There were some issues that came up in the exploratory analysis that required attention. First, it was necessary to eliminate the extreme outliers that had been observed. I also explored other options to handle outliers as I suspect they are one of the key problems with this dataset that affects modelling. Isolation Forest which uses a random forest model to identify outliers seemed like an effective systematic solution. However, when implemented, the model identified 1007 outliers which was far too large a number. As well, the exact nature of the outliers would be unknown unless significant examination was undertaken. It is possible that adjusting the model’s parameters could have reduced the number of outliers but it still seemed like too extreme a solution especially when the dataset was relatively small. Instead, I focused on outliers in the target variable on the assumption that very expensive or very cheap properties would befuddle prediction models. I found that removing properties with a sale price in the 1st and 99th percentiles – just over 60 records – significantly improved model performance. This does, however, come at a cost of a degree of information loss and resulting models might have difficulty with predicting extremely high or low property values. The second step involved a bit of feature engineering. After a tight fit between combined Total Above Ground Living Area Square Footage and Total Basement Square Footage and Sale Price, I decided to combine the two features to create a Total Square Footage variable. The exploratory analysis also revealed that there were a lot of levels in nominal variables such as Neighborhood and some of these had very few records. As such, I set about reducing levels by consolidating sparsely populated levels for some of these variables. For Neighborhood, I spent quite a bit of time studying the map of neighborhoods in Ames in order to see if I could fit some of the neighborhoods with fewer values into larger neighborhoods. This proved quite effective as I was able to consolidate many tiny neighborhoods into adjoining neighborhoods that share a high level of similarity reducing total Neighborhoods from 28 to 15. As well, I did some less involved consolidation for Foundation, Exterior 1st and Condition 1(identifying the proximity of a property to roadways, railroads and amenities). The third step involved performing several operations on either parts of the dataset or the whole dataset. For example, in order to approximate a normal distribution, a log1p transformation was performed on the target variable as well as numerical features that met a skew threshold of 0.5. Unlike scaling or normalization, this transformation simply applies a log function to the data and did not involve training on the whole dataset that might create bias in the test data. The probability plot shows that it was quite effective in normalizing SalePrice.

Chart, line chart

Description automatically generated

One-hot encoding was performed on categorical variables in order to make them useable during modelling. The data was then partitioned into train and test sets using a ratio of 70% to 30%. After this, the test and train data were standardized in order to help reduce any distortions that might occur due to large differences in the scale of the features. This was explicitly done after data partitioning in order to prevent any data leakage whereby information from outside the training set is used to fit the models thereby creating bias.

I found that it was possible to leave the test data unscaled and generate results that were similar to those generated using standardized test data.[[3]](#footnote-3) The problem was that leaving the test data unscaled distorted several charts that displayed linear relationships and residual plots. It is true that scaling additional newly collected data would be burdensome but given that scaling this new data, is strictly speaking, not necessary this should not prove a problem. As well, X\_train data was standardized using fit.transform while X\_test was standardized with transform alone which should ensure that there was no data leakage.

#### Feature Selection

For feature selection, a few different methods were utilized. First, I selected a basic set of 16 features selected using the variables that are most closely correlated with Sale Price minus those that exhibit significant multicollinearity. These variables include many of those that one would expect to be good predictors of price such as Overall Quality and Total Above Grade Living Area.

|  |  |
| --- | --- |
| Basic Features: | |
| Overall Quality | Full Bathrooms |
| Above Ground Living Area Square Footage | Year Remodel/Addition |
| External Quality | Fireplace Quality |
| Total Basement Area Square Footage | Masonry Veneer Area |
| Garage Area | Heating Quality |
| Basement Quality | Basement Finish 1 Square Footage |
| Year Built | Lot Frontage |
| Garage Finish | Lot Area |

The next type of feature selection uses a random forest model. The model uses the average decrease in variance (i.e. mean square error) that is achieved by adding a feature to the model to identify the best predictors. Using the random forest selector with the default hyperparameters results in the selection of 16 variables.

A linear model using these variables did not produce results that were substantially different from the basic feature selection. This is not surprising as the random forest selector picked several of the same features as were contained in the basic set. Through adjusting the random forest model’s hyperparameters, such as maximum tree depth and number of predictors, a much larger set of features was selected. As is noted below in the section on preliminary models, a linear model with this larger feature set was able to outperform a linear model that used the basic feature set.

Chart

Description automatically generated

The last type of feature selection uses a Lasso regression model. The Lasso model is a type of regularized regression which applies a penalty equal to the absolute value of the magnitude of the coefficients. Of particular importance in relation to feature selection, Lasso reduces the coefficients with those variables least important to the model decreased to zero or near zero. In this manner, the model chooses the most important variables. In relation to the Ames data, the Lasso selector initially chose 0 variables with the significance threshold set to 0.25. As the threshold is reduced to 0.0199 the number of variables selected expands to 20. A model with these select features produced fairly good results. Similar to the results of hyperparameter tuning the random forest selector, adjusting the model’s threshold downward produces a larger number of features. By testing the results with a linear model and adjusting the threshold accordingly, a set of 61 features was found to be an optimal number.

Chart

Description automatically generated

Both of the advanced methods of feature selection were able to pick a large number of features that can be used via linear regression to provide an effective model. It is interesting, though, that features selected by the two models were very different. On one hand, the random forest model heavily favored numerical (particularly continuous) variables and included many of the top correlated features among its top choices. On the other hand, the Lasso selector included a large number of nominal variables and many of these were among those with the highest coefficients.

#### Scoring – RMSE, MAE and R2

All of the models developed for this study, were scored using k-fold cross validation scorers for root mean squared error (RMSE), mean absolute error (MAE) and adjust R2 . Both RMSE and MAE provide a measure of the size of the residuals - the difference between the values predicted by the model and the observed values. The advantage of RMSE is, given that it involves squaring residuals, it is much more sensitive to large values. In contrast, MAE offers a measure that is more stable and insensitive to outliers. Adjusted R2 which indicates that percentage of total variance captured by the model (minus a penalty for the number of features) offers a more relative measure that can be used in spite of changes in the scale of the data.

To measure the efficiency of the models, the time required to fit the model was measured. This was implemented using Python’s timeit function which takes the average time after looping over the model seven times.

Finally, the stability of the models was assessed by looking at changes in predictive performance that occur when the k number of folds is changed. The base k value used was 10 with 5, 15, 20 and 25 providing variations.

#### Preliminary Models

At this stage of the project, I created a number of multiple linear regression models for the dataset. These models serve two key purposes. First, they provide a baseline in terms of the metrics that are being used to assess the effectiveness of models in the study RMSE, MAE and adjusted R2. The second purpose of the linear models is to provide some comparison of the different sets of features selected by our three different feature selectors – basic correlation, random forest, Lasso regression.

Initially, I tried to create a multiple regression model using all 82 features. This, however, proved problematic. The assumption was that this “everything included” model would provide a relatively high score in terms of RMSE and MAE which would serve as a baseline. However, when all the features were included, the model was effectively broken with a RMSE of 5750750202 and an adjusted R2 of -4.75867E+18. Cleary something was very wrong with this model.

Chart, scatter chart

Description automatically generated

As a result, the base model was a multiple linear regression model using the basic feature selection consisting of the 16 variables that were most highly correlated with sale price. The results of this model were quite good with an RMSE score of 0.1192 on the test data.

The second linear model used the features that were identified by the random forest feature selector. The model which incorporated 62 features offered a slight improvement on the previous basic feature model. It produced a RMSE of 0.1044, a MAE of 0.079 and an adjusted R2 of 0.9074 or the test data. It is notable, though, that the stability of this model was somewhat problematic.

The third linear model used the features that were identified by the Lasso feature selector. This model provided the best RMSE, MAE and adjusted R2 scores - respectively 0.0975, 0.0714, and 0.9203. That said, the results of all 3 sets of features are fairly consistent.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Linear Regression | LR BasicFeatures | LR RForestFeat Select | LR LassoFeat Select |
| AdjR2 CV Test | -1.12693E+21 | 0.880479444 | 0.907446193 | 0.920278763 |
| AdjR2 CV Train | -4.75867E+18 | 0.889505338 | 0.907278511 | 0.925615478 |
| MAE CV Test | 658335239.5 | 0.088441631 | 0.078827173 | 0.071427533 |
| MAE CV Train | 21855952.04 | 0.090010453 | 0.081881244 | 0.072035191 |
| RMSE CV Test | 5750750202 | 0.119216204 | 0.104427186 | 0.097481367 |
| RMSE CV Train | 300810559.9 | 0.12346723 | 0.112765123 | 0.10098801 |

Rather than simply using one of the three sets of features, I created a set of 40 features which combines elements from all three sets.

|  |  |  |
| --- | --- | --- |
| Main Feature Set: | | |
| Total SF | Heating QC | Garage Finish |
| Overall Qual | Overall Cond | Central Air |
| Year Built | Kitchen Qual | Neighborhood(15) |
| Garage Area | Exter Qual | Foundation (3) |
| Year Remod/Add | BsmtFin SF 1 | Sale Condition\_Abnormal |
| Mas Vnr Area | Sale Type\_New | Full Bath |
| Full Bath | Sale Condition\_Family | Exterior 1st\_BrkFace |
| Bsmt Qual | Exterior 1st\_AsbShng | Condition 1\_PosN |
| Lot Area |  |  |

# Modelling

After completing a number of preliminary models, the project moves into the final modelling phase. A total of five different models were developed in addition to linear regression: 1. Lasso Regression, 2. Decision Tree Regressor, 3. Random Forest Regressor, 4. Gradient Boosting (XGBoost), 5. Artificial Neural Network Perceptron (Keras).

## Multiple Linear Regression

The multiple linear regression model was included here so as to have a linear model that uses the same feature set as all other models. This actually proved somewhat problematic. When some slightly larger feature sets were used, the multiple linear model again broke as it did when all features were included. However, the linear model performs extremely well with the feature set that was settled upon. Indeed, it is somewhat surprisingly one of the top performing models finishing marginally behind Lasso Regression in terms of model effectiveness while clearly outperforming in terms of efficiency. Although there are, in fact, stability issues with the linear model, in the case of this particular feature set, there was no large difference as the number of k folds was changed during cross-validation.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Linear Regression |  |  |  |  |  |  |
|  | **K5** | **K10** | **K15** | **K20** | **K25** | **Min/Max%Change** |
| AdjR2 CV Test | 0.908602 | **0.904509** | 0.902587 | 0.90258 | 0.900686 | 1% |
| AdjR2 CV Train | 0.904913 | **0.905381** | 0.904345 | 0.904499 | 0.903117 | 0% |
| MAE CV Test | 0.081444 | **0.08071** | 0.080405 | 0.08054 | 0.08074 | 1% |
| MAE CV Train | 0.08362 | **0.083152** | 0.083177 | 0.082934 | 0.083061 | 1% |
| RMSE CV Test | 0.106899 | **0.10596** | 0.105744 | 0.10522 | 0.105075 | 2% |
| RMSE CV Train | 0.115234 | **0.113837** | 0.113623 | 0.113309 | 0.113181 | 2% |

## Lasso Regression CV

Least Absolute Shrinkage and Selection Operator (Lasso) regression is an L1 form of regularized regression. As was mentioned in the feature selection section, the model works by setting a constant that is equal to the sum of the absolute magnitude of the coefficients and reducing the coefficients of various features with the least import coefficients dropping to zero. The effect is to slightly increase the model’s bias in order to achieve a reduction in variance. In general, regularization tends to be very useful when the number of observations in a dataset is close to or even less than the number of dimensions. As such, Lasso is particularly well suited to the Ames data where there are a high number of features relative to a more modest number of observations.

The results show that, as expected, Lasso regression is very effective. In fact, it is the best model in terms of all effectiveness metrics, very narrowly surpassing Multiple Linear Regression. On the other hand, it performs significantly worse than plain linear regression in terms of time efficiency but this is likely largely down to the cross-validation that is built into the model. It is also exceedingly stable.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lasso Regression |  |  |  |  |  |  |
|  | **K5** | **K10** | **K15** | **K20** | **K25** | **Min/Max%Change** |
| AdjR2 CV Test | 0.909596 | **0.905079** | 0.903205 | 0.903158 | 0.901238 | 1% |
| AdjR2 CV Train | 0.905007 | **0.905401** | 0.904461 | 0.904578 | 0.90313 | 0% |
| MAE CV Test | 0.080917 | **0.080352** | 0.080017 | 0.080132 | 0.080451 | 1% |
| MAE CV Train | 0.083516 | **0.083043** | 0.083034 | 0.082822 | 0.082934 | 1% |
| RMSE CV Test | 0.10631 | **0.10559** | 0.105376 | 0.104836 | 0.104782 | 1% |
| RMSE CV Train | 0.115175 | **0.113817** | 0.113553 | 0.113259 | 0.113155 | 2% |

Chart, scatter chart

Description automatically generated

## The Decision Tree Regressor

The Decision Tree Regressor was included because both random forest and the variety of gradient boosting used in this study both incorporate regression tree teachers at the basic starting point for their models. As mentioned in the literature review, regression trees rely on a process of recursive partitioning where features and weightings are used to split the dataset according to which combination results in the greatest reduction in variance (mean absolute error)(Zurda 2011). Such a process allows regression trees to effectively model data with non-linear characteristics that would prove problematic to a basic linear model. However, regression trees also suffer from some serious drawbacks. In particular, by selecting one independent feature at a node over others, the regression tree can de-emphasize aspects of the dataset. This can also contribute to model instability. Finally, the regression tree model is prone to overfitting if its branches are not limited or pruned.

For this model, as well as all the other machine learning models, I used a set procedure. After a little bit (or a lot) of trial and error, I utilized scikitlearn’s RandomizedSearchCV to more systematically tune these hyperparameters. Initially, I was using large values for tree depth. However, when I graphed the loss function I found that the RMSE declined very sharply until a tree depth of approximately 8 at which point it jumped up and down before stabilizing at a depth of about 20.

Graphical user interface, chart, application

Description automatically generated

The graphic below shows the first 3 of the tree’s 8 levels.[[4]](#footnote-4) We can see that the initial split is on Overall Quality that is less than or equal to 0.284. It includes all records (2003) and results in a reduction of mean squared error of 0.14. In subsequent splits we can see that Total SF is often the deciding feature with other splits on Year Built, Year Remodel/Add, Basement Quality and Lot Area.Diagram

Description automatically generated

The results demonstrate that the Decision Tree Regressor was the poorest performing model in the study in terms of effectiveness. It managed only 0.1695 RMSE, 0.1267 MAE, and 0.7597 Adjusted R2 on the test data. The model, though, is quite efficient. It has one of the lowest time-costs at an average 8.118ms per loop of any of models in the study except for plain Multiple Regression.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Regression Tree |  |  |  |  |  |  |
|  | **K5** | **K10** | **K15** | **K20** | **K25** | **MinMax%Change** |
| AdjR2 CV Test | 0.770807 | **0.759655** | 0.769479 | 0.781301 | 0.776348 | 3% |
| AdjR2 CV Train | 0.819382 | **0.827071** | 0.824896 | 0.828529 | 0.826834 | 1% |
| MAE CV Test | 0.126194 | **0.126762** | 0.121433 | 0.119243 | 0.121842 | 6% |
| MAE CV Train | 0.11664 | **0.112462** | 0.114263 | 0.112172 | 0.112407 | 4% |
| RMSE CV Test | 0.169448 | **0.169452** | 0.161157 | 0.158376 | 0.159896 | 7% |
| RMSE CV Train | 0.158743 | **0.154893** | 0.15564 | 0.15301 | 0.152712 | 4% |

Though its results were fairly uniform, the Decision Tree Regressor was comparatively one of the more unstable models in terms of results changing as the k increased for cross-validation.

Chart, scatter chart

Description automatically generated

## Random Forest Regressor

With the next model, we move to the Random Forest Regressor which. as the name suggests, is actually a forest of regression trees. As an ensemble model, random forest draws on many of the strengths of the regression tree while also addressing that model’s weaknesses (Mayer 2019). In particular, by using a large number of trees, bagging (sampling with replacement) and randomization of features considered for splitting, the random forest generally creates a strong predictor that captures (i.e reduces) considerable variation in the data while avoiding the problem of overfitting. The Random Forest Regressor model was expected to be amongst the better performing models in the study.

Using a similar approach to that of the Regression Tree, after a period of trial and error, I set up a parameter grid and ran RandomizedSearchCV to tune the parameters. Originally, I assumed that the hyperparameters would be virtually identical to the settings for the random forest feature selector as had been the case with the Lasso model. This turned out to be incorrect and resulted in quite poor performance. While the feature selector had worked best with very shallow trees and an excessive number of trees, the Random Forest Regressor model needed significantly deeper trees along with a smaller number of trees to obtain better results. With random forest, figuring out how to chart the loss function was a bit more complicated. Rather than just charting the decline in RMSE as the number of estimators (i.e. trees) increase, I utilized Out-of-Bag (OOB) scoring. This method utilizes the average error for each datapoint using predictions from trees that do not contain that particular datapoint (Oob errors for random forests, scikit-learn). Visualizing the rate of OOB errors relative to the number of trees in the forest, revealed that the optimal forest size was 315 trees.

Chart, histogram

Description automatically generated

Below is a visualization that shows the first 3 levels of the 315th tree in the random forest. This particular tree uses a sample of 1250 observations from the 2003. The first split is based on Basement Quality with subsequent splits dictated by a variety of different features from Foundation type to Central Air to Neighborhood (Iowa DOT & Railroad). Though the tree has 50 levels, we can see that, already on the 3rd level, one branch ends with two leaf nodes. This tree is evidently very different than that of the Decision Tree Regressor model. This is as expected given that Random Forest’s bagging and randomized feature split selection decorrelates the various trees to capture (i.e. reduce) a greater amount of the variation in the data.

Diagram

Description automatically generated

The results of the random forest model were quite good with a RMSE of 0.1241 and a MAE of 0.0885 on the test data. This performance is not at all terrible and within the range of the best performing models. Nevertheless, the random forest model was still outperformed by a linear regression model that simply used highly correlated variables. What was particularly interesting is that the Random Forest Regressor actually underperformed a linear model that used features selected via a random forest selector. This suggests that the feature selection element of the random forest model was sound, but that regression tree models were perhaps particularly ill-suited to the particular data set. The results of the random forest were especially disappointing as it is also quite time inefficient. Random forest was a very stable model only slightly bested by Lasso and Multiple Linear Regression.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Random Forest |  |  |  |  |  |  |
|  | **K5** | **K10** | **K15** | **K20** | **K25** | **Min/Max%Change** |
| AdjR2 CV Test | 0.874026 | **0.872414** | 0.873766 | 0.870991 | 0.87307 | 0% |
| AdjR2 CV Train | 0.891569 | **0.893476** | 0.892141 | 0.892884 | 0.892105 | 0% |
| MAE CV Test | 0.090392 | **0.088548** | 0.088679 | 0.08855 | 0.089384 | 2% |
| MAE CV Train | 0.087067 | **0.085935** | 0.086431 | 0.085859 | 0.085529 | 2% |
| RMSE CV Test | 0.12579 | **0.124122** | 0.122661 | 0.122586 | 0.121685 | 3% |
| RMSE CV Train | 0.123072 | **0.121109** | 0.121454 | 0.120457 | 0.120181 | 2% |

## Gradient Boosting (XGBoost)

Like random forest, Gradient Boosting is also an ensemble model that uses regression trees as its base learner (Mayer 2019). The key difference is that this model focuses on an additive approach using iterative optimization (i.e. gradient descent) to minimize the loss function (i.e. mean squared error) at each step. The basic idea is to take a large number of weak learners together to form a strong learning model. These weak learners here are relatively shallow regression trees. The initial tree is fit to the data. The model then readjusts to place more weight on the datapoints that the model had trouble predicting. Each successive tree then becomes an expert on a slice of the data so when averaged together they form a strong predictor. In the literature review, Gradient Boosting was the model that most often provided the best housing predictions so we would expect it to perform well in this study. XGBoost is popular and highly optimized implementation of gradient boosting that also includes L1 regularization.

With XGBoost, I found that I had to modify the method that I followed with the other machine learning models. I had tried to come up with a fairly fleshed out model using a combination of trial and error and a systematic evaluation using RandomizedSearchCV. However, when I tried to run a prediction of this intermediate model in order to visualize the loss function, memory usage exploded with Windows crashing moments later. As such, I began by charting a simple model in order to visualize the decline in RMSE relative to the number of estimators. My earlier experimentation with XGBoost flowed from the assumption that the number of estimators would be quite large - roughly comparable to the optimal number of trees in the random forest model. The visualization made clear that this was quite mistaken as the optimal number of boosting rounds was under 20. Nevertheless, in practice I found that somewhat better results could be achieved by bumping the number of estimators up to 100.

Chart, line chart

Description automatically generated

As expected, XGBoost is one of the best performing models overall. While it very slightly bested the Multiple Linear Regression and Lasso models on all measures of effectiveness for the training data, it was slightly worse with the actual test data. Even so, the difference in scores for training and test data was very small indicating that overfitting is not a major problem. In terms of efficiency, as an ensemble model it did require more time than the Decision Tree Regressor, Lasso, and Multiple Linear Regression models. Once the number of estimators was reduced, XGBoost did outperform the random forest model. In terms of stability, the XGBoost model had no significant problems.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| XGBoost |  |  |  |  |  |  |
|  | **K5** | **K10** | **K15** | **K20** | **K25** | **Min/Max%Change** |
| AdjR2 CV Test | 0.892685 | **0.892479** | 0.891062 | 0.887362 | 0.887816 | 1% |
| AdjR2 CV Train | 0.905558 | **0.908649** | 0.905533 | 0.9072 | 0.902803 | 1% |
| MAE CV Test | 0.08604 | **0.083682** | 0.082785 | 0.084133 | 0.083399 | 4% |
| MAE CV Train | 0.081933 | **0.08027** | 0.081531 | 0.080386 | 0.081308 | 2% |
| RMSE CV Test | 0.116106 | **0.112065** | 0.111791 | 0.11306 | 0.111282 | 4% |
| RMSE CV Train | 0.114797 | **0.111700** | 0.112952 | 0.111594 | 0.112925 | 3% |

Chart, scatter chart

Description automatically generated

## Artificial Neural Network - Perceptron (Keras)

The ANN Perceptron is a mathematical model used in supervised learning that resembles a simplified version of a network of neurons in the human brain. The model is typically made up of several layers of interconnected nodes with the first layer representing inputs to the model, the last layer representing outputs, and the connection between nodes representing weightings (Grus, 2015; Kirk, 2017; Limsombunc,2004). In the case of a classification problem, each interior node in the network has an activation function. Like a neuron, it fires if a threshold is met and passes the piece of data on to the next part of the network, or alternatively does not fire with the piece of data stopping at that point. In contrast, for a regression problem, each of the interior nodes represents a linear equation supplied with inputs and weights, and the activation function represents a potential transformation of the data before moving on to the next part of the network. One can think of a simple 0 layer network with an input and output layer as equivalent to a multiple linear equation. The number of input nodes represents the number of independent variables in equation, the connections represent the coefficient weights, and the single output node represents the equation and result. The weights of the model are initially set randomly and are adjusted as the model is iteratively trained. Data is fed through the model, the results are compared with the training data, and then the weightings are adjusted for each subsequent run. Increasing the number of interior nodes and layers gives the model expanded capacity to handle increased complexity in the data (e.g. non-linearity).

As has been discussed, ANN perceptron models were some of the machine learning models used in housing price prediction, but the performance of these models has been somewhat mixed. In our case, the sophistication of the ANN model is likely ill-suited to our simple dataset (i.e. analytical overkill). Nevertheless, it was interesting to see how the model performed.

Implementing the ANN model was considerably more difficult than the other models. *It should be noted that I utilized a GPU in order to speed things along which means that the model had an advantage in terms of time testing.* The first challenge was designing the neural network itself. The input layer was straightforward as it is set at the number of features in the dataset and the output layer is a single node. The questions that were difficult to answer were – how many intermediate (hidden) layers should the model include and how many nodes should be in each layer? It was quite surprising that there does not seem to be definite rules governing this element of creating a neural network model. Based on anecdotal information, I started with a model that had a single hidden layer with roughly double the number of nodes as independent variables in the dataset. Below is a simple visualization that shows part of the overall model.

A picture containing diagram

Description automatically generated

Once again, I set about systematically testing various sizes of the single intermediate layer along with the optimal settings for epoch and batch size. This proved somewhat problematic. Even with RandomizedSearchCV and a very limited number of possible settings, it took a considerable amount of time to conduct a test run. As well, despite there being no problems running ANN models individually, attempting to run cross-validation optimization almost immediately used up all available memory. The solution, to decrease the number of CPU cores available and limiting the number of tasks being queued, further slowed things down. As a result, my experimenting with different network designs was truncated. I did find, nevertheless, that a network with double the nodes in the hidden layer as the input layer performed quite well. On the other hand, adding a second hidden layer with about half the number of nodes as the first hidden layer performed more poorly than the single layer model. As a result, further testing was limited to the single layer model though it may be that a multi-intermediate layer model could possibly improve performance.

Chart, histogram

Description automatically generated

Checking the loss function of the ANN model was also not straightforward. Graphing the decline in RMSE with the number of epochs, showed incredible results with RMSE values below 0.025 for both the training and the test set with less than 25 epochs. However, running the model based on these settings through the cross-validation scorer, resulted in significantly worse results with a RMSE train score of 0.14 and a RMSE test score of over 0.2. This might be a result of an error in implementation or coding, but the Keras regressor does have a scikit learn wrapper than makes it quite similar to other models. It could also be due to the fact that each fold of cross validation contains significantly less data than the whole test set and this relative shortage of data resulted in underperformance.

Ultimately, I found that a model with a single-hidden layer of 85 nodes trained with 100 epochs and a batch size of 25 produced the best results. Unfortunately, as the number of cross-validation rounds was increased to 25, the model became unstable and it took several attempts to get a result. Even with L1 regularization to resolve this issue, I was not able to create a significantly more stable model at this k level.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Neural Network |  |  |  |  |  |  |
|  | **K5** | **K10** | **K15** | **K20** | **K25** | **Min/Max%Change** |
| AdjR2 CV Test | 0.807707 | 0.821829 | 0.82658 | 0.831866 | 0.830482 | 3% |
| AdjR2 CV Train | 0.867103 | 0.872189 | 0.873703 | 0.869683 | 0.877158 | 1% |
| MAE CV Test | 0.116163 | 0.106024 | 0.104806 | 0.099281 | 0.100072 | 17% |
| MAE CV Train | 0.100577 | 0.09726 | 0.096518 | 0.097013 | 0.095418 | 5% |
| RMSE CV Test | 0.152511 | 0.143286 | 0.144096 | 0.135601 | 0.144228 | 12% |
| RMSE CV Train | 0.135511 | 0.132625 | 0.131956 | 0.132926 | 0.129656 | 5% |

The results make clear what had been suspected: an ANN model is not the most appropriate method for the Ames Housing dataset. The Perceptron was the second worst performing model behind only the Decision Tree Regressor model. That said its effectiveness results were not terrible and within a reasonable distance of better performing models. However, the efficiency cost in terms of processing time was by far the highest of any model even using the additional processing power of a graphic processing unit. It also proved to be the least stable of the models.

## Modelling Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Effectiveness  Test Data @10K CV | | Efficiency  Avg 7 Runs | Stability  Avg Min/Max%Change on All Effectiveness Metrics |
| Multiple Linear Regression | RMSE | 0.10596 | 2.21ms | 1% |
| MAE | 0.08071 |
| Adj R2 | 0.904509 |
| Lasso Regression | RMSE | 0.10559 | 28.1ms | 1% |
| MAE | 0.080352 |
| Adj R2 | 0.905079 |
| Decision Tree Regressor | RMSE | 0.169452 | 8.13ms | 4% |
| MAE | 0.126762 |
| Adj R2 | 0.759655 |
| Random Forest Regressor | RMSE | 0.124122 | 239ms | 2% |
| MAE | 0.088548 |
| Adj R2 | 0.872414 |
| Gradient Boosting (XGBoost) | RMSE | 0.112065 | 115ms | 2% |
| MAE | 0.083682 |
| Adj R2 | 0.892479 |
| Artificial Neural Network | RMSE | 0.143286 | 12600ms | 7% |
| MAE | 0.106024 |
| Adj R2 | 0.821829 |

# Conclusions

This study set out to examine the performance of machine learning models when applied to the problem of predicting housing prices. As was discussed in the literature review, predicting property prices using analytical techniques is not at all new. Hedonic housing models that utilize multiple regression have a long history. This then frames the key research questions that this study addressed.

***Are more novel machine learning methods a significant improvement on traditional regression methods and, if so, in what ways?***

Due to the results of the study, it is necessary to answer the secondary research question before the primary research question. In the case of the Ames dataset, the results of this study show that linear multiple regression particularly in its advanced Lasso form are actually superior to the machine learning models in terms of effectiveness, efficiency and stability. Such a comparison must, however be kept firmly in context – the context of this specific dataset. It was a relatively small dataset with less than 3000 observations and significant portions of the data displayed fairly strong linear relationships to the target variable. Lasso regression was particularly well-suited to the problem of a large number of features. It is really striking how well plain linear regression performed. In this particular application, one could possibly argue that no more complex model was really needed. Nevertheless, it must be noted that the multiple regression model could not handle the dataset when all features were included and it performed best when supplied with a selection of features using more advanced methods. That said, the results here show that one must be careful about assuming that the newest and most advanced techniques in the data science arsenal are superior in every case. There are still and likely will be for some time, use cases that clearly call for a multiple regression model.

***Which current analytical methods and machine learning techniques perform best in predicting housing prices?***

Despite the strong performance of linear regression, the results also showed that machine learning models can certainly be effective in predicting housing prices. All the models investigated here produced strong predictive results and even the poorest performers were not extremely far off from the results of the best performing models. Gradient Boosting via XGBoost proved to be especially effective when compared to other machine learning models. It had the strongest performance in terms of all the measures of effectiveness on both the training and test sets. It also scored impressively in terms of efficiency even though it is a more complex ensemble model. In contrast, the Decision Tree Regressor had the worst effectiveness scores but was the top performer aside from the plain linear model in terms of efficiency. The Random Forest Regressor model was, somewhat surprisingly, one of the poorer performing models finishing just behind the stronger performers in terms of effectiveness and very clearly one of the worst performers in terms of time efficiency. The Artificial Neural Network model also performed quite poorly though its results improved dramatically after some optimization.

Again, all these results must be kept in context. In this instance, advanced machine learning models were ill suited to the use case. Machine learning models such as Random Forest, Gradient Boosting and especially Artificial Neural Networks would likely perform better when supplied with much, much larger datasets. As well, the real strengths of these advanced models is that they are able to do precisely what linear models struggle with – deal effectively with non-linear relationships in the data. In many cases, one would want to utilize one of these more advanced models given that linearity cannot easily be assumed given that these models perform quite well on linear data and excel when the data is more complex.

At first glance, the findings here seem to run counter to much of our literature review which presented a strong case that machine learning models were generally better than traditional hedonic models relying on linear regression. However, aside from being a bit too dismissive of linear regression (especially for simple scenarios), the real conclusion of the literature review was that there is no one single best machine learning model in terms of housing price prediction. This fits well with the results presented in this study. It is true that Gradient Boosting is generally a very strong performer. It is also true that Artificial Neural Network models are at the cutting edge of all kinds of interesting and innovative techniques using a wide variety of different kinds of data. On the other hand, in spite of being less high powered in terms of price prediction, the time efficiency of regression trees along with their ability to handle non-linear data might make them the ideal choice if one is looking for a simple model capable of processing tons of data with a low overhead cost.

The key conclusion then is that context matters. A Ferrari isn’t the best vehicle to pick up the kids from school and Louboutin’s are not great for hiking. Overeager data science students would do well to remember that predicting house prices in Ames, Iowa hardly requires an Artificial Neural Network. Beyond this obvious conclusion, it remains the case that trying a variety of different models on a dataset is a very important part of the overall data science project process, and one should be careful about making assumptions about best-performing models without such practical testing.

This study also set out to reproduce the results of a study of the Ames dataset by Singh and his co-authors. The findings of this study in relation to the performance of Random Forest Regressor and Gradient Boosting models are very similar. There is a significant difference, however, in the treatment of the basic linear regression model. Singh and his co-authors were able to implement an “all-features included” multiple linear regression model for the Ames data. While their model did not break, the results were very poor. As such, they found that the Random Forest model and Gradient Boosting were by far the superior methods for housing price prediction. My findings, though, cast some doubt on this conclusion as I found that a multiple linear regression model that used a simple traditional method of feature selection, like correlation, could provide much better results. It might be suggested then that, by not implementing a more realistic linear regression model, Singh and his co-authors set up multiple linear regression as something of a strawman that is easily knocked down by more advanced techniques. This was perhaps not deliberate on their part but rather simply a matter of expediency as they were really interested in the results produced by the machine learning models.

This study had a variety of limitations and shortcomings which must be acknowledged. First, and perhaps most importantly, the housing price prediction presented here is extremely time-bound. The models only provide predictions of house prices in Ames, Iowa between 2006 and 2010. They do not take into account long or medium-term trends in the housing market. Further, the US financial crisis and the associated downturn in the housing market during this period, the worst since the Great Depression, suggests that this was an exceptional moment rather than a typical period that represents a normal housing market. This is not to suggest that the type of short-term price prediction presented here is without value. Instead, the increased ability to access large amounts of data with greater and greater detail in real time means that short-term price prediction is an increasingly important tool. Nevertheless, such prediction is only effective if its limitations are made plain. A second shortcoming, as previously mentioned, was the relatively small size of the dataset that very likely hindered some of the machine learning models. Third, beyond some very broad references to neighborhood, the models presented here did not incorporate location or geographical information which is an important aspect of modern models of housing price prediction. While efforts were certainly made to fine tune the advanced machine learning models utilized in this study, it is very possible that further improvement gains could be achieved through more skillful optimization. Finally, the analysis presented here only looks at one half of price determination. That is to say it looks only at the demand-side without incorporating determinants of housing supply.

There are several areas where future research could extend this study. First, one could replicate this study using an updated version of the Ames dataset. This would be quite feasible given that the Ames City Assessor Office offers much of this data via their on-line portal. Second, it might be possible to conduct a study across several communities similar in size and demographics to Ames. A larger project might attempt to provide a comparison of different housing price prediction models in several communities of different sizes (e.g. small, medium, and large). Finally, it would be interesting to develop a project that processes real time real estate data to compare different methods of housing price prediction.

# Dataset Features

### Dependent Variable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Name | Description | Data Type | Mean | Standard Deviation | Mode | Min | 25 | Median | 75 | Max | Skew | Kurtosis | Na |
| 1 | Sale Price | Sale price of property | int64 | 180796.06 | 79886.692 | 135000 | 12789 | 129500 | 160000 | 213500 | 755000 | 1.744 | 5.119 | 0 |

### Discrete Variables

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Name | Description | Data Type | Mean | Standard Deviation | Mode | Min | 25 | Median | 75 | Max | Skew | Kurtosis | Na |
| 1 | Bsmt Full Bath | Full bathrooms in basement | float64 | 0.431 | 0.525 | 0 | 0 | 0 | 0 | 1 | 3 | 0.617 | -0.748 | 2 |
| 2 | Bsmt Half Bath | Half bathrooms in basement | float64 | 0.061 | 0.245 | 0 | 0 | 0 | 0 | 0 | 2 | 3.941 | 14.922 | 2 |
| 3 | Full Bath | Full bathrooms above grade | int64 | 1.567 | 0.553 | 2 | 0 | 1 | 2 | 2 | 4 | 0.172 | -0.541 | 0 |
| 4 | Half Bath | Half bathrooms above grade | int64 | 0.380 | 0.503 | 0 | 0 | 0 | 0 | 1 | 2 | 0.698 | -1.030 | 0 |
| 5 | Bedroom AbvGr | Bedrooms above grade | int64 | 2.854 | 0.828 | 3 | 0 | 2 | 3 | 3 | 8 | 0.306 | 1.891 | 0 |
| 6 | Kitchen AbvGr | Kitchens above grade | int64 | 1.044 | 0.214 | 1 | 0 | 1 | 1 | 1 | 3 | 4.314 | 19.870 | 0 |
| 7 | TotRms AbvGrd | Total rooms above grade (not inc. bathrooms) | int64 | 6.443 | 1.573 | 6 | 2 | 5 | 6 | 7 | 15 | 0.754 | 1.155 | 0 |
| 8 | Fireplaces | Number of fireplaces | int64 | 0.599 | 0.648 | 0 | 0 | 0 | 1 | 1 | 4 | 0.739 | .102 | 0 |
| 9 | Garage Cars | Car capacity of garage | float64 | 1.767 | 0.761 | 2 | 0 | 1 | 2 | 2 | 5 | -0.220 | 0.245 | 1 |

### Continuous Variables

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Name | Description | Data Type | Mean | Standard Deviation | Mode | Min | 25 | Median | 75 | Max | Skew | Kurtosis | Na |
| 1 | Lot Frontage | Square footage of property connected to street | float64 | 69.225 | 23.365 | 60 | 21 | 58 | 68 | 80 | 313 | 1.499 | 11.235 | 490 |
| 2 | Lot Area | Size of property in square feet | int64 | 10147.922 | 7880.018 | 9600 | 1300 | 7440.25 | 9436.5 | 11555.250 | 215245 | 12.821 | 265.024 | 0 |
| 3 | Mas Vnr Area | Area of masonry veneer in square feet | float64 | 101.897 | 179.113 | 0 | 0 | 0 | 0 | 164 | 1600 | 2.607 | 9.287 | 23 |
| 4 | BsmtFin SF 1 | Area of the primary basement finish type in square feet | float64 | 442.630 | 455.591 | 0 | 0 | 0 | 370 | 734 | 5644 | 1.416 | 6.859 | 1 |
| 5 | BsmtFin SF2 | Area of secondary basement finish type in square feet | float64 | 49.722 | 169.168 | 0 | 0 | 0 | 0 | 0 | 1526 | 4.140 | 18.781 | 1 |
| 6 | Bsmt Unf SF | Area of unfinished basement in square feet | float64 | 559.263 | 439.494 | 0 | 0 | 219 | 466 | 802 | 2336 | 0.923 | 0.410 | 1 |
| 7 | Total Bsmt SF | Total area of basement in square feet | float64 | 1051.615 | 440.615 | 0 | 0 | 793 | 990 | 1302 | 6110 | 1.156 | 9.136 | 1 |
| 8 | 1st Flr SF | Area of first floor in square feet | int64 | 1159.558 | 391.891 | 864 | 334 | 876.250 | 1084 | 1384 | 5095 | 1.469 | 6.969 | 0 |
| 9 | 2nd Flr SF | Area of second floor in square feet | int64 | 335.456 | 428.396 | 0 | 0 | 0 | 0 | 703.750 | 2065 | 0.866 | -0.415 | 0 |
| 10 | Low Qual Fin SF | Total area of low quality finish in square feet | int64 | 4.677 | 46.311 | 0 | 0 | 0 | 0 | 0 | 1064 | 12.118 | 175.607 | 0 |
| 11 | Gr Liv Area | Total above ground living area in square feet | int64 | 1499.690 | 505.509 | 864 | 334 | 1126 | 1442 | 1742.750 | 5642 | 1.274 | 4.138 | 0 |
| 12 | Garage Area | Garage area in square feet | float64 | 4672.820 | 215.047 | 0 | 0 | 320 | 480 | 576 | 1488 | 0.242 | 0.951 | 1 |
| 13 | Wood Deck SF | Wood deck area in square feet | int64 | 93.752 | 126.362 | 0 | 0 | 0 | 0 | 168 | 1424 | 1.843 | 6.754 | 0 |
| 14 | Open Porch SF | Open porch area in square feet | int64 | 47.533 | 67.483 | 0 | 0 | 0 | 27 | 70 | 742 | 2.535 | 10.954 | 0 |
| 15 | Enclosed Porch | Enclosed porch area in square feet | int64 | 23.012 | 64.139 | 0 | 0 | 0 | 0 | 0 | 1012 | 4.104 | 28.487 | 0 |
| 16 | 3Ssn Porch | 3-season porch area in square feet | int64 | 2.592 | 25.141 | 0 | 0 | 0 | 0 | 0 | 508 | 11.404 | 149.989 | 0 |
| 17 | Screen Porch | Screen porch area in square feet | int64 | 16.002 | 56.087 | 0 | 0 | 0 | 0 | 0 | 576 | 3.957 | 17.859 | 0 |
| 18 | Pool Area | Pool areas in square feet | int64 | 2.243 | 35.597 | 0 | 0 | 0 | 0 | 0 | 800 | 16.939 | 299.775 | 0 |
| 19 | Misc Val | Dollar value of miscellaneous features | int64 | 50.635 | 566.344 | 0 | 0 | 0 | 0 | 0 | 17000 | 22.00 | 566.203 | 0 |

### Ordinal Variables

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Name | Description | Data Type | # of Levels | Levels/Instances | Na |
| 1 | Lot Shape | Shape of property | object | 4: Regular, IR1(Slightly Irregular), IR2(Moderately Irregular), IR3(Irregular) | Reg(1859), IR1(979), IR2(76), IR3(16) |  |
| 2 | Utilities | Type of Utilities available | object | 3: All Public(AllPub), Electric/Gass/Septic Tank(NoSewr), Electric/Gas(NoSeWa), Electric(ELO) | AllPub(2927),NoSewr(2),NoSeWa(1) |  |
| 3 | Land Slope | Slope of property | object | 3: Gentle slope(Gtl), Moderate Slope (Mod), Severe Slope(Sev) | Gtl(2789), Mod(125), Sev(16) |  |
| 4 | Overall Qual | Rating overall materials/finish of home | in64 | 10: Very excellent(10), Excellent(9), Very Good(8), Good(7), Above Average(6), Average(5), Below Average(4), Fair(3), Poor(2), Very Poor(1) | 5(825),6(732),7(602),8(350),4(226), 9(107), 3(40), 10(31), 2(13), 1(4) |  |
| 5 | Overall Cond | Rating overall condition of home | int64 | 10: Very excellent(10), Excellent(9), Very Good(8), Good(7), Above Average(6), Average(5), Below Average(4), Fair(3), Poor(2), Very Poor(1) | 5(1654), 6(533), 7(390), 8(144), 4(101), 3(50), 9(41), 2(10), 1(7) |  |
| 6 | Exter Qual | Rating quality of exterior materials | object | 5: Excellent(Ex), Good(Gd), Typical(TA), Fair(Fa), Poor (Po) | TA(1799), Gd(989), Ex(107), Fa(35) |  |
| 7 | Exter Cond | Rating condition of exterior of home | object | 5: Excellent(Ex), Good(Gd), Typical(TA), Fair(Fa), Poor (Po) | TA(2549), Gd(299), Fa(67), Ex(12), Po(3) |  |
| 8 | Bsmt Qual | Rating height of basement | object | 5: Excellent 100 => inches(Ex), Good 90-99 (Gd), Typical 80-89(TA), Fair 70-79(Fa), Poor <70(Po) | TA(1283), Gd(1219), Ex(258), Fa(88), Po(2) | 80 |
| 9 | Bsmt Cond | Rating general condition basement | object | 5: Excellent(Ex), Good(Gd), Typical(TA), Fair(Fa), Poor(Po) | TA(2616), Gd(122), Fa(104), Po(5), Ex(3) | 80 |
| 10 | Bsmt Exposure | Refers to walk out or garden level walls | object | 4: Good Exposure(Gd), Average Exposure (Av), Minimum Exposure(Mn), No Exposure(No) | No(1906), Av(418), Gd(284), Mn(239) | 83 |
| 11 | BsmtFin Type 1 | Rating primary basement finish | object | 6:Good Living Quarters(GLQ), Average Living Quarters(ALQ), Below Average Living Quarters(BLQ), Average Rec Room(Rec), Low Quality Finish(LwQ), Unfinished(Unf) | GLQ(859), Unf(851), ALQ(429), Rec(288), BLQ(269), LwQ(154) | 80 |
| 12 | BsmtFin Type 2 | Rating secondary basement finish | object | 6:Good Living Quarters(GLQ), Average Living Quarters(ALQ), Below Average Living Quarters(BLQ), Average Rec Room(Rec), Low Quality Finish(LwQ), Unfinished(Unf) | Unf(2499), Rec(106), LwQ(89), BLQ(68), ALQ(53), GLQ(34) | 81 |
| 13 | HeatingQC | Rating heating quality | object | 5: Excellent(Ex), Good(Gd), Typical(TA), Fair(Fa), Poor (Po) | Ex(1495), TA(864), Gd(476), Fa(92), Po(3) |  |
| 14 | Electrical | Rating electrical system | object | 6: Standard Circuit Breaker/Romex(SBrkr), Fuse Box over 60 AMP/Romex(FuseA), 60 AMP Fuse Box/Mostly Romex(FuseF), 60 AMP Fuse Box/Knob&Tube(FuseP), Mixed(Mix) | SBrkr(2682), FuseA(188), FuseF(50), FuseP(8), Mix(1) | 1 |
| 15 | KitchenQual | Rating kitchen quality | object | 5: Excellent(Ex), Good(Gd), Typical(TA), Fair(Fa), Poor (Po) | TA(1494), Gd(1160), Ex(205), Fa(70), Po(1) |  |
| 16 | Functional | Functionality of home | object | 8: Typical Functionality(Typ), Minor Deductions 1(Min1), Minor Deductions(Min2), Moderate Deductions(Mod), Maj1(Major Deductions 1), Maj2(Major Deductions 2), Severely Damaged(Sev), Salvage Only(Sal) | Typ(2728), Min2(70), Min1(65), Mod(35), Maj1(19), Maj2(9), Sev(2), Sal(2) |  |
| 17 | FireplaceQu | Rating fireplace | object | 5: Exceptional Masonry(Ex), Masonry Main Level(Gd), Prefab Main Level(TA), Prefab Basement(Fa), Ben Franklin Stove(Po) | Gd(744), TA(600), Fa(75), Po(46), Ex(43) | 1422 |
| 18 | Garage Finish | Interior finish of garage | object | 3: Finished(Fin), Rough Finished(RFn), Unfinished (Unf) | Unf(1231), RFn(812), Fin(728) | 159 |
| 19 | Garage Qual | Rating quality of garage | object | 5: Excellent(Ex), Good(Gd), Typical(TA), Fair(Fa), Poor (Po) | TA(2615), Fa(124), Gd(24), Po(5), Ex(3) | 159 |
| 20 | Garage Cond | Rating condition of garage | object | 5: Excellent(Ex), Good(Gd), Typical(TA), Fair(Fa), Poor (Po) | TA(2665), Fa(74), Gd(15), Po(14), Ex(3) | 159 |
| 21 | Paved Drive |  | object | 3: Paved(Y), Partial(P), Dirt/Gravel(N) | Y(2652), N(216), P(62) |  |
| 22 | Pool QC | Rating of pool quality | object | 4: Excellent(Ex), Good(Gd), Typical(TA), Fair(Fa) | Ex(4), Gd(4) TA(3), Fa(2) | 2917 |
| 23 | Fence | Rating of Fence Quality | object | 4: Good Privacy(GdPrv), Minimum Privacy(MnPrv), Good Wood(GdWo), Minimum Wood/Wire(MnWw) | MnPrv(330), GdPrv(118), GdWo(112), MnWw(12) | 2358 |

**Nominal Variables**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Name | Description | Data Type | Levels | Categories [Code](Instances) | NA |
| 1 | MS SubClass | Identifies type of dwelling | int64 | 16 | 1Story 1946-Newer[020](1079), 1Sto 1945-Older[030](139), 1Sto FinishedAttic All Ages[040](6), 1-1/2Sto Unfinished AllA[045](18), 1-1/2Sto Finished AllA[050](287), 2Sto 1946-Newer[060](575), 2Sto 1945-Older[070](128), 2-1/2Sto All Ages[075](23), Split/MultiLevel[080](118), Split Foyer [085](48), Duplex [090](109), 1-Story **P**lanned **U**nit **D**evelopment 1946-Newer [120](192), 1-1/2Sto PUD All Ages [150](1), 2Sto PUD 1946-Newer [160](129), PUD MultiLevel[180](17), 2 Family Conversion [190](61) |  |
| 2 | MS Zoning | Identifies zoning class | object | 7 | Agriculture[A](2), Commerical[C](25), Floating Village[FV](139), Industrial[I](2), Residential High Density[RH](27), Residential Low Density[RL](2273), Residential Low Density/Park[RP], Residential Medium Density[RM](462) |  |
| 3 | Street | Type of road access | object | 2 | Gravel[Grvl](12), Paved[Pave](2918) |  |
| 4 | Alley | Type of alley access | object | 2 | Gravel[Grvl](120), Paved[Pave](78) | 2732 |
| 5 | Land Contour | Flatness of property | object | 4 | NearFlat/Level[Lvl](2633), Banked[Bnk](117), Hillside[HLS](120), Depression [Low](60) |  |
| 6 | Lot Config | Configuration of lot | object | 5 | Inside lot[Inside](2140), Corner Lot[Corner](511), CulDSac[Cul-de-sac](180), Frontage 2 sides [FR2](85), Frontage 3 sides[FR3](14) |  |
| 7 | Neighborhood | General location in City of Ames | object | 28 | Bloomington Heights[Blmngtn](28), Bluestem[Blueste](10), Briardale[BrDale](30), Brookside[BrkSide](108), Clear Creek[ClearCr](44), College Creek[CollgCr](267), Edwards(194), Gilbert(165), Greens(8), Green Hills[GrnHill](2), Iowa DOT and Rail Road [IDOTRR](93), Landmark[Landmrk](1), Meadow Village[MeadowV](37), Mitchell(114), North Ames[Names](443), Northridge[NoRidge](71), Northpark Villa[NPkVill](23), Northridge Heights[NridgHt](166), NorthwestAmes[NWAmes](131), OldTown(239), South/West of Iowa State U[SWISU](48), Sawyer(151), Sawyer West[SawyerW](125), Somerset(182), Stone Brook [StoneBr](51), Timberland [Timber](72), Veenker(24) |  |
| 8 | Condition 1 | Proximity to miscellaneous conditions | object | 9 | **Adj**acent arterial street[Artery](92), Adj feeder street[Feedr](164), Normal[Norm](2522), Within 200’ North-South Railroad [RRNn](9), Adj to North-South Railroad [RRAn](50), Near positive offsite feature [PosN](20), Adj positive offsite feature[PosN](39), Within 200’ East-West Railroad [RRNe](6), Adj East-West Railroad[RRAe](28) |  |
| 9 | Condition 2 | Proximity to secondary conditions | object | 8 | **Adj**acent arterial street[Artery](5), Adj feeder street[Feedr](13), Normal[Norm](2900), Within 200’ North-South Railroad [RRNn](2), Adj to North-South Railroad [RRAn](1), Near positive offsite feature [PosN](4), Adj positive offsite feature [PosA](4), Within 200’ East-West Railroad [RRNe](0), Adj East-West Railroad[RRAe](1) |  |
| 10 | Bldg Type | Type of home | object | 5 | Single-Family Detached[1Fam](2425), Two-Family Conversion[2FmCon](62), Duplex[Duplx](109), Townhouse End Unit[TwnhsE](233), Townhouse Inside Unit[Twnhs](101) |  |
| 11 | House Style | Style of home | object | 8 | One Story[1Story](1481), One & ½ Sto 2nd level finished [1.5Fin](314), One & ½ Sto 2nd level unfinished[1.5Unf](19), Two Sto [2Story](873), Two and ½ story 2nd level finished[2.5Fin](8), Two Sto and ½ Sto 2nd level unfinished[2.5Unf](24), Split Foyer[SFoyer](83), Split Level [SLvl](128) |  |
| 12 | Roof Style | Type of roof | object | 6 | Flat(20), Gable(2321), Barn[Gambrel](22), Hip(551), Mansard(11), Shed(5) |  |
| 13 | Roof Matl | Roof material | object | 8 | Clay or Tile [ClyTile](1), Standard Shingle[CompShg](2887), Membrane[Membran](1), Metal(1), Roll(1), Gravel & Tar [Tar&Grv](23), Wood Shakes[WdShake](9), Wood Shingles[WdShngl](7) |  |
| 14 | Exterior 1 | Exterior covering on house | object | 16 | Abestos Shingles[AsbShng](44), Asphalt Shingles[AsphShn](44), Brick Common[BrkComm](6), Brick Face[BrkFace](88), Cinder Block[CBlock](2), Cement Board[CemntBd](126), Hard Board[Hdboard](442), Imitation Stucco[ImStucc](1), Metal siding[MetalSd](450), Other(0), Plywood(221), PreCast(1), Stone(2), Stucco(43), Vinyl Siding[VinylSd](1026), Wood Siding[Wd Sdng](420), Wood Shingles[WdShing](56) |  |
| 15 | Exterior 2 | Secondary exterior covering on house | object | 17 | Abestos Shingles[AsbShng], Asphalt Shingles[AsphShn], Brick Common[BrkComm], Brick Face[BrkFace], Cinder Block[CBlock], Cement Board[CemntBd], Hard Board[Hdboard](406), Imitation Stucco[ImStucc](15), Metal siding[MetalSd](447), Other(1), Plywood(274), PreCast(1), Stone(6), Stucco(47), Vinyl Siding[VinylSd](1015), Wood Siding[Wd Sdng](397), Wood Shingles[WdShing](81) |  |
| 16 | Mas Vnr Type | Type of mason veneer | object | 5 | Brick Common[BrkCmn](25), Brick Face[BrkFace](880), Cinder Block[CBlock](1), None(1752), Stone(249) | 23 |
| 17 | Foundation | Type of foundation | object | 6 | Brick & Tile [BrkTil](311), Cinder Block[CBlock](1244), Poured Concrete [PConc](1310), Slab(49), Stone(11), Wood(5) |  |
| 18 | Heating | Type of heating | object | 6 | Floor Furnace[Floor](1), Gas Forced Warm Air Furnace[GasA](2885), Gas Hot Water or Steam [GasW](27), Gravity Furnace [Grav](9), Hot Water or Steam Heat Non-Gas [OthW](2), Wall Furnace [Wall](6) |  |
| 19 | Central Air | Central air conditioning | object | 2 | Yes[Y](2734), No[N](196) |  |
| 20 | Garage Type | Garage location | object | 6 | More than one type [2Types](23), Attached[Attchd](1731), Basement[Basment](36), BuiltIn(186), CarPort(15), Detached[Detchd](782) | 157 |
| 21 | Misc Feature | Miscellaneous features not included in other categories | object | 5 | Elevator[Elev](1), 2nd Garage[Gar2](5), Other[Othr](4), Shed(95), Tennis Court[TenC](1) | 2824 |
| 22 | Sale Type | Type of sale | object | 10 | Warranty Deed[WD](2536), Warranty Deed Cash[CWD](12), Warranty VA Loan[VWD](1), Home just constructed[New](239), Court Officer Deed/Estate [COD](87), Contract 15% Down Regular Term[Con](5), Contract Low Down Payment Low Interest[ConLw](8), Contract Low Interest[ConLI](9), Contract Low Down Payment[ConLD](26), Other[Oth](7) |  |
| 23 | Sale Condition | Condition of sale | object | 6 | Normal Sale [Normal](2413), Abnormal Sale[Abnorml](190), Adjoining Land[AdjLand](12), Allocation – linked properties[Alloca](24), Sale between family members[Family](46), Home not completed when last assessed[Partial](245) |  |

# Data Appendix

### Effectiveness Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K5 | Linear Regression | Lasso Regression | Regression Tree | Random Forest | XGBoost | NeuralNet |
| **AdjR2 CV Test** | 0.908602261 | 0.90959559 | 0.770806958 | 0.874025809 | 0.892684543 | 0.807706956 |
| **AdjR2 CV Train** | 0.904912819 | 0.90500744 | 0.819381858 | 0.891569009 | 0.905557966 | 0.86710259 |
| **MAE CV Test** | 0.081443678 | 0.080916994 | 0.126193598 | 0.090391524 | 0.086040427 | 0.116162786 |
| **MAE CV Train** | 0.083619724 | 0.083516011 | 0.116640183 | 0.087066652 | 0.081933386 | 0.100576903 |
| **RMSE CV Test** | 0.106898593 | 0.106310453 | 0.169447709 | 0.125789957 | 0.116106087 | 0.15251077 |
| **RMSE CV Train** | 0.115233981 | 0.11517456 | 0.158743257 | 0.123072357 | 0.114796747 | 0.135510764 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| K10 | **Linear Regression** | **Lasso Regression** | **Regression Tree** | **Random Forest** | **XGBoost** | **NeuralNet** |
| **AdjR2 CV Test** | 0.90450863 | 0.905078797 | 0.759655126 | 0.872413587 | 0.892478645 | 0.821828635 |
| **AdjR2 CV Train** | 0.905381214 | 0.90540125 | 0.827071262 | 0.893476254 | 0.908648653 | 0.872189326 |
| **MAE CV Test** | 0.080710214 | 0.080352347 | 0.126761962 | 0.088548393 | 0.083682152 | 0.106024205 |
| **MAE CV Train** | 0.083151699 | 0.083043375 | 0.112461763 | 0.085935163 | 0.080269942 | 0.097259547 |
| **RMSE CV Test** | 0.105960152 | 0.105589799 | 0.169452231 | 0.124122333 | 0.112064975 | 0.143286449 |
| **RMSE CV Train** | 0.113836999 | 0.113817226 | 0.154893182 | 0.121109419 | 0.111700002 | 0.132625064 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| K15 | **Linear Regression** | **Lasso Regression** | **Regression Tree** | **Random Forest** | **XGBoost** | **NeuralNet** |
| **AdjR2 CV Test** | 0.902586618 | 0.903205036 | 0.769479377 | 0.873765614 | 0.891062312 | 0.826579621 |
| **AdjR2 CV Train** | 0.904344559 | 0.904460827 | 0.824895574 | 0.89214084 | 0.905533281 | 0.873702838 |
| **MAE CV Test** | 0.080404725 | 0.08001737 | 0.121432512 | 0.088678778 | 0.082785066 | 0.104805526 |
| **MAE CV Train** | 0.083176948 | 0.083033523 | 0.114262916 | 0.086430865 | 0.081530638 | 0.096517644 |
| **RMSE CV Test** | 0.105743828 | 0.105375609 | 0.161157283 | 0.122660683 | 0.111790642 | 0.144095701 |
| **RMSE CV Train** | 0.1136226 | 0.113553234 | 0.155639948 | 0.121453895 | 0.112952158 | 0.131955922 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| K20 | **Linear Regression** | **Lasso Regression** | **Regression Tree** | **Random Forest** | **XGBoost** | **NeuralNet** |
| **AdjR2 CV Test** | 0.902579524 | 0.90315772 | 0.781300982 | 0.870991265 | 0.887361574 | 0.831865695 |
| **AdjR2 CV Train** | 0.904498806 | 0.904577593 | 0.828529168 | 0.892884104 | 0.907199635 | 0.869682937 |
| **MAE CV Test** | 0.080539984 | 0.080131655 | 0.119243389 | 0.088550474 | 0.08413301 | 0.099280673 |
| **MAE CV Train** | 0.08293444 | 0.082822206 | 0.112171656 | 0.085859454 | 0.080385998 | 0.097012934 |
| **RMSE CV Test** | 0.105220406 | 0.104836195 | 0.158376419 | 0.122585633 | 0.113060239 | 0.135601333 |
| **RMSE CV Train** | 0.113308826 | 0.113258621 | 0.153010215 | 0.120457092 | 0.111593545 | 0.132925785 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| K25 | **Linear Regression** | **Lasso Regression** | **Regression Tree** | **Random Forest** | **XGBoost** | **NeuralNet** |
| **AdjR2 CV Test** | 0.900685899 | 0.901237806 | 0.776348362 | 0.873070427 | 0.887816084 | 0.830482434 |
| **AdjR2 CV Train** | 0.903117062 | 0.903130349 | 0.826834422 | 0.892104727 | 0.902803337 | 0.877158469 |
| **MAE CV Test** | 0.080740075 | 0.08045063 | 0.121841751 | 0.089384148 | 0.083398965 | 0.100072339 |
| **MAE CV Train** | 0.083060804 | 0.082933527 | 0.112407256 | 0.085529049 | 0.081307861 | 0.095417735 |
| **RMSE CV Test** | 0.105075223 | 0.104782063 | 0.159896445 | 0.121684834 | 0.111282319 | 0.14422759 |
| **RMSE CV Train** | 0.113180944 | 0.113154891 | 0.152712358 | 0.120181287 | 0.112925312 | 0.129656062 |

### Results with No Standardization of Test Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Linear Regression | Lasso Regression | Regression Tree | Random Forest | XGBoost | NeuralNet |
| **AdjR2 CV Test** | 0.904509 | 0.903784 | 0.759436 | 0.872258 | 0.892467 | 0.018325 |
| **AdjR2 CV Train** | 0.905381 | 0.905401 | 0.827071 | 0.893476 | 0.908649 | 0.879124 |
| **MAE CV Test** | 0.08071 | 0.081063 | 0.126875 | 0.088533 | 0.083695 | 0.190698 |
| **MAE CV Train** | 0.083152 | 0.083043 | 0.112462 | 0.085935 | 0.08027 | 0.097747 |
| **RMSE CV Test** | 0.10596 | 0.106296 | 0.169509 | 0.124198 | 0.112071 | 0.368755 |
| **RMSE CV Train** | 0.113837 | 0.113817 | 0.154893 | 0.121109 | 0.1117 | 0.133105 |

### Efficiency Results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Linear Regression | Lasso Regression | Regression Tree | Random Forest | XGBoost | NeuralNet |
| Trial 1 | 2.07 | 26.2 | 8.08 | 248 | 110 | 11.7 |
| Trial 2 | 2.21 | 28.1 | 8.13 | 239 | 115 | 12.6 |
| Trial 3 | 1.94 | 25.8 | 8.04 | 239 | 107 | 11.5 |
| Trial 4 | 1.87 | 31.9 | 8.15 | 239 | 114 | 12.2 |
| Trial 5 | 1.83 | 32.9 | 8.19 | 242 | 125 | 12.6 |
| Avg | 1.984 | 28.98 | 8.118 | 241.4 | 114.2 | 12.12 |

### Hardware Comparison

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ryzen 5 5600X  (6 Cores 32GB) | Apple M1  (8 cores 16GB) | Intel Remote Virtual Machine  (2 Cores 8GB) |
| Linear Regression | 1.984ms | 4.38ms | 5.49ms |
| Lasso | 28.98ms | 63.3ms | 54.6ms |
| Regression Tree | 8.118ms | 9.43ms | 27.8ms |
| Random Forest | 241.4ms | 357ms | 1.48s |
| XGBoost | 114.2ms | 725ms | 857ms |

# References

Abdulhafedh, A. (2022). Incorporating multiple linear regression in predicting the house prices using a big real estate dataset with 80 independent variables. *OALib*, *09*(01), 1–21. https://doi.org/10.4236/oalib.1108346

Boelaert, J., & Ollion, E. (2018). The Great Regression. Machine Learning, Econometrics, and the Future of Quantitative Social Sciences . *Revue Française De Sociologie*, *59*(3), 475–506.

Čeh, M., Kilibarda, M., Lisec, A., & Bajat, B. (2018). Estimating the performance of Random Forest versus multiple regression for predicting prices of the apartments. *ISPRS International Journal of Geo-Information*, *7*(5), 168. https://doi.org/10.3390/ijgi7050168

Colwell, P. F., & Dilmore, G. (1999). Who was first? an examination of an early hedonic study. *Land Economics*, *75*(4), 620–626. https://doi.org/10.2307/3147070

De Cock, D. (2011). Ames, Iowa: Alternative to the Boston Housing Data as an end of semester regression project. *Journal of Statistics Education*, *19*(3). https://doi.org/10.1080/10691898.2011.11889627

Do, A. Q., & Grudnitski, G. (1992). A Neural Network Approach to Residential Property Appraisal. The Real Estate Appraiser, 39–45.

Embaye, W. T., Zereyesus, Y. A., & Chen, B. (2021). Predicting the rental value of houses in household surveys in Tanzania, Uganda and Malawi: Evaluations of hedonic pricing and machine learning approaches. *PLOS ONE*, *16*(2). https://doi.org/10.1371/journal.pone.0244953

Fan, G.-Z., Ong, S. E., & Koh, H. C. (2006). Determinants of house price: A decision tree approach. *Urban Studies*, *43*(12), 2301–2315. https://doi.org/10.1080/00420980600990928

Goodman, A. C., & Thibodeau, T. G. (2003). Housing market segmentation and hedonic prediction accuracy. *Journal of Housing Economics*, *12*(3), 181–201. https://doi.org/10.1016/s1051-1377(03)00031-7

Grus, J. (2015). Data Science from scratch: First principles with Python. O'Reilly Media.

Haider, M. (2016). *Getting started with data science: Making sense of data with analytics*. IBM.

Haider, Murtaza (1999). *Development of hedonic prices indices for freehold property in the greater Toronto area: Application of spatial autoregressive techniques* [Masters of Applied Science, University of Toronto. TSpace]

Hausler, J., Ruscheinsky, J., & Lang, M. (2018). News-based sentiment analysis in real estate: A machine learning approach. *Journal of Property Research*, *35*(4), 344–371. https://doi.org/10.1080/09599916.2018.1551923

Ho, W. K. O., Tang, B.-S., & Wong, S. W. (2020). Predicting property prices with machine learning algorithms. *Journal of Property Research*, *38*(1), 48–70. https://doi.org/10.1080/09599916.2020.1832558

Hoepner, A. G., McMillan, D., Vivian, A., & Wese Simen, C. (2020). Significance, relevance and explainability in the machine learning age: An econometrics and Financial Data Science Perspective. *The European Journal of Finance*, *27*(1-2), 1–7. https://doi.org/10.1080/1351847x.2020.1847725

Hong, J., Choi, H., & Kim, W.-sung. (2020). A house price valuation based on the random forest approach: The mass appraisal of residential property in South Korea. *International Journal of Strategic Property Management,* 24(3). Retrieved February 14, 2022, from https://doi.org/10.3846/ijspm.2020.11544

Ja’afa, N. S., Mohamad, J., & Ismail, S. (2021). Machine learning for property price prediction and price valuation: A systematic literature review. *PLANNING MALAYSIA: Journal of the Malaysian Institute of Planners*, *19*(3), 411–422.

Kirk, M. (2017). *Thoughtful machine learning with python: A test-driven approach*. O'Reilly.

Kontrimas, V., & Verikas, A. (2011). The mass appraisal of the real estate by Computational Intelligence. *Applied Soft Computing*, *11*(1), 443–448. https://doi.org/10.1016/j.asoc.2009.12.003

Limsombunc, V., Gan, C., & Lee, M. (2004). House price prediction: Hedonic price model vs. Artificial Neural Network. *American Journal of Applied Sciences*, *1*(3), 193–201. https://doi.org/10.3844/ajassp.2004.193.201

Mayer, M., Bourassa, S. C., Hoesli, M., & Scognamiglio, D. (2019). Estimation and updating methods for hedonic valuation. *Journal of European Real Estate Research*, *12*(1), 134–150. https://doi.org/10.1108/jerer-08-2018-0035

McCluskey, W. J., McCord, M., Davis, P. T., Haran, M., & McIlhatton, D. (2013). Prediction accuracy in mass appraisal: A comparison of modern approaches. *Journal of Property Research*, *30*(4), 239–265. https://doi.org/10.1080/09599916.2013.781204

Oob errors for random forests. scikit-learn. (n.d.). Retrieved April 4, 2022, from https://scikit-learn.org/stable/auto\_examples/ensemble/plot\_ensemble\_oob.html

Özsoy, O., & Şahin, H. (2009). Housing price determinants in Istanbul, Turkey. *International Journal of Housing Markets and Analysis*, *2*(2), 167–178. https://doi.org/10.1108/17538270910963090

Peng, T.-C. (2021). The effect of hazard shock and disclosure information on property and land prices: A machine-learning assessment in the case of Japan. *Review of Regional Research*, *41*(1), 1–32. https://doi.org/10.1007/s10037-020-00148-1

Poursaeed, O., Matera, T., & Belongie, S. (2018). Vision-based real estate price estimation. *Machine Vision and Applications*, *29*(4), 667–676. https://doi.org/10.1007/s00138-018-0922-2

Rodney Ramcharan (March, 2006). Regressions: Why Are Economists Obessessed with Them? *Finance & Development* 43(1). https://www.imf.org/external/pubs/ft/fandd/2006/03/basics.htm

Selim, H. (2009). Determinants of house prices in Turkey: Hedonic regression versus artificial neural network. *Expert Systems with Applications*, *36*(2), 2843–2852. https://doi.org/10.1016/j.eswa.2008.01.044

Simlai, P. (2017). Spatial dependence, idiosyncratic risk, and the valuation of disaggregated housing data. *The Journal of Real Estate Finance and Economics*, *57*(2), 192–230. https://doi.org/10.1007/s11146-017-9610-7

Singh, A., Sharma, A., & Dubey, G. (2020). Big data analytics predicting real estate prices. *International Journal of System Assurance Engineering and Management*, *11*(S2), 208–219. https://doi.org/10.1007/s13198-020-00946-3

Sirmans, G. S., Macpherson, D., & Zietz, E. N. (2005). The Composition of Hedonic Pricing Models. *Journal of Real Estate Literature*, *13*(5), 3–43.

Tay, D. P. H. & Ho, D. K. H. (1992). Artificial Intelligence and the mass appraisal of residential apartments. Journal of Property Valuation and Investment, 10(2), 525–540. https://doi.org/10.1108/14635789210031181

Tchuente, D., & Nyawa, S. (2021). Real estate price estimation in French cities using geocoding and machine learning. *Annals of Operations Research*, *308*(1-2), 571–608. https://doi.org/10.1007/s10479-021-03932-5

Valier, A. (2020). Who performs better? avms vs hedonic models. *Journal of Property Investment & Finance*, *38*(3), 213–225. https://doi.org/10.1108/jpif-12-2019-0157

Wang, P.-Y., Chen, C.-T., Su, J.-W., Wang, T.-Y., & Huang, S.-H. (2021). Deep learning model for house price prediction using heterogeneous data analysis along with joint self-attention mechanism. *IEEE Access*, *9*, 55244–55259. https://doi.org/10.1109/access.2021.3071306

Worzala, E., Lenk, M., & Silva, A. (1995). An exploration of neural networks and its application to real estate valuation. *Journal of Real Estate Research*, *10*(2), 185–201. https://doi.org/10.1080/10835547.1995.12090782

Xin, S. J., & Khalid, K. (2018). Modelling House Price Using Ridge Regression and Lasso Regression. *International Journal of Engineering & Technology*, *7*(4.3), 498–501.

Xu, L., & Li, Z. (2020). A new appraisal model of second-hand housing prices in China’s first-tier cities based on machine learning algorithms. *Computational Economics*, *57*(2), 617–637. https://doi.org/10.1007/s10614-020-09973-5

Zurada, J., Levitan, A., & Guan, J. (2011). A comparison of regression and artificial intelligence methods in a mass appraisal context. *Journal of Real Estate Research*, *33*(3), 349–388. https://doi.org/10.1080/10835547.2011.12091311

1. Project Github Repository: https://github.com/ddsarai/CIND820Ames [↑](#footnote-ref-1)
2. There are, however, a number of conference papers related to Ames dataset that come up on Google Scholar. Unfortunately, I was unable to obtain any of them. [↑](#footnote-ref-2)
3. See Data Appendix – below main results [↑](#footnote-ref-3)
4. For full tree please see code notebook. [↑](#footnote-ref-4)