University of Zurich

Department of Economics

Machine Learning for Economic Analysis

Prof. Damian Kozbur

Matteo Courthoud

Second Project:

Predicting Housing Prices

Group:

Samir Sulejmani 16-726-747

Felix Jost 15-067-861

Marc Zeugin 15-708-993

Simon Cappelli 11-945-813

08.01.2021

Table of contents

1. Introduction

Accurately predicting residential house prices can be beneficial for a number of stakeholders. Developers and investors are better able to determine the returns for their projects and can better analyze the risk and rewards associated with the projects. Accurate predictions are also beneficial for tax assessment and for real estate appraisal, as they allow for an easy and cost-efficient method to determine the value of a property. Additionally, these predictions can be used by insurances in order to more precisely calculate premiums that have to be paid for home insurance as well as the potential losses associated with the insured house. Policy makers and urban planners can profit from house price prediction as it allows them to define city areas into different segments using appropriate zoning schemes. It can also be used by prospective homeowners to compare the consumption value of the house and the risks related to such an investment. The goal of this paper is to create an accurate prediction of the residential house prices with the available data.

In this paper we compare the results of OLS regression with Lasso and Ridge regression, Random Forest, Gradient Boosting and Neural Networks in order to predict the residential housing prices in Ames, Iowa from 2006 to 2010 as accurately as possible. The price for a specific house can be determined by a number of factors including the location of the house, neighborhood, the condition of the house, the size of the house and the land as well as other factors. The data used in this paper is provided by the Kaggle Challenge "House Prices – Advanced Regression Techniques". The dataset provides housing data with 79 explanatory variables describing different aspects of residential homes in Ames, Iowa. The complete list of all 79 explanatory variables can be found in the Appendix. The dataset is split into a training dataset, which can be used to build and develop the models and the testing dataset, where the built models can be tested.

This paper is structured as follows: Section 2 first provides an overview of the preparation and cleaning of the data, then an introduction to the machine learning methods used in this paper as well as the implementation are presented. In Section 3 the results will be shown, and we will compare the results of the OLS regression with the results of the Ridge Regression, Lasso Regression, Random Forest, Gradient Boosting, Deep Neural Networks as well as Hybrid Predictions.

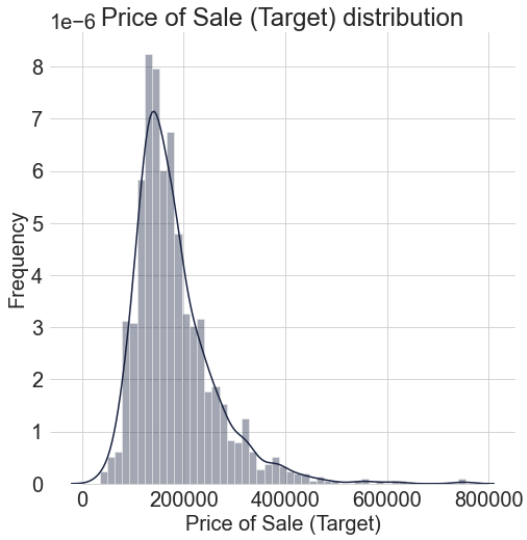
2. Methodology and Implementation

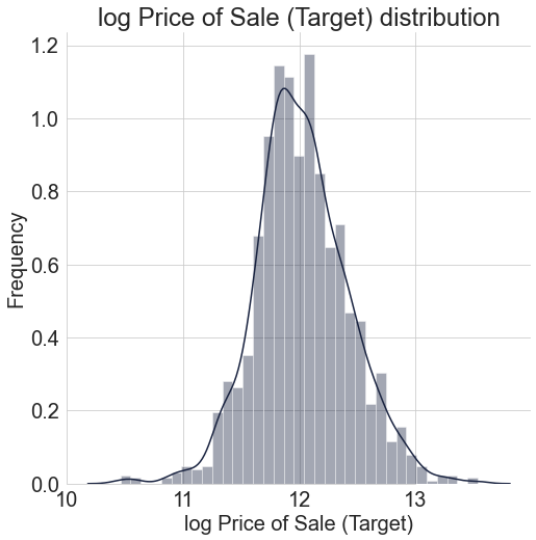
2.1. Data Cleaning and Exploration

The Kaggle challenge "House Prices – Advanced Regression Techniques" provides 1460 training observations with sales prices which served as our dataset for this project. The data contains 79 explanatory variables which provide information on a wide variety of factors about the properties themselves, their locations and various aspects of their associated buildings. 19 of the explanatory variables are continuous, 14 discrete and 46 categorical.

Initial examination of the sales prices revealed a positively skewed distribution. Therefore, we used logarithmic transformation which yielded an approximately normally distributed target variable. After the logarithmic transformation it became apparent that there were still some outliers. Since residential properties are often sold below market price within families, it might be beneficial to drop observations with particularly low sales prices. Similarly, very expensive properties are often associated with famous former owners or a historical legacy. Such intangible

factors can lead to inflated sales prices at a specific moment. But these influences may not be stable over time and are unrelated to the fundamental value of the property. Thus, an insurance company may wish to drop some high-priced observations to generate more stable predictions while a real-estate investor might want to capture those effects to identify profitable resale opportunities. Additionally, any chosen cutoff value is necessarily arbitrary. Therefore, we estimated all our models twice, once with the complete dataset and once without outliers in the sales price variable. To identify outliers, we defined the minimum value as the first quartile minus 1.5 times the interquartile range and the maximum value as the third quartile plus 1.5 times the interquartile range. Observations with sales prices outside this range were dropped for the calculations without outliers.





In addition to this, we needed to address missing values in the explanatory variables. Most missing values correspond to variables that were not applicable to the specific observation. For example, the quality rating of a basement area is necessarily unavailable for a house without a basement. As basement quality in our dataset is a categorical variable, missing values were interpreted as a separate category where there was no basement. Similarly, most missing values represented a separate category indicating that the feature in question was not present on a property. However, with larger numbers of missing values this strategy risks serious bias if there are actually missing values where the variable would be applicable. Therefore, we decided to drop variables with more than 2% missing values. This was applicable to 13 out of the 79 variables. Fitting the models without dropping those variables yielded more unstable and less accurate predictions which confirmed that no significant explanatory power was lost. In all other cases, we replaced missing values with the most frequent category for categorical variables. Missing numerical values were replaced with the mean value of the variable.

Finally, we converted the categorical variables into dummies, which yielded a training dataset with 1460 observations and 255 explanatory variables. We further split this set using 67% of the observations to train our models and the remaining 33% as testing sample to assess model performance. This was done by calculating the root mean squared error (RMSE) of a model's predictions for the holdout sample observations. It is important to note that the RMSE of the models OLS and Neural Networks did differ quite substantially between the random state of the

training and testing split. For this reason, we show all model scores in Chapter 3 both with and without outliers, as this gives us some intuition if our results are robust or not.

2.2. OLS Regression

We used ordinary least squares regression as a baseline model to predict the natural logarithm of the sales prices using all 255 explanatory variables. This minimizes the following formula in the training set:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2$$

Where y_i is the logarithmic sales price of the house in question, β_0 is the constant, and $\beta_j x_{ij}$ represents the OLS coefficients multiplied with the respective input data.

Our OLS model achieved a test RMSE of 0.184 with outliers and a test RMSE 0.166 without outliers. Naturally, OLS is prone to overfitting and predictions may vary a lot depending on the specific split into training and holdout data. It is also sensitive to collinearity issues which will arise in this dataset as there are several related explanatory variables. The following methods aim to generate more accurate and reliable predictions.

2.3. Ridge Regression

Ridge Regression functions similar to least squares, instead of seeking coefficient estimates that best fit the data by minimizing RSS, Ridge Regression applies a so-called shrinkage penalty in form of $\lambda \sum_{j=1}^{p} \beta_j^2$. Thus, the goal of Ridge Regression is to minimize the following formula:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2 = RSS + \lambda \sum_{j=1}^{p}\beta_j^2$$

The shrinkage parameter λ, also called ridge estimator or tuning parameter, shrinks the estimates of β_j towards zero. This is called L2 regularization and will not set any parameter to zero, just shrink them towards zero.

For the Ridge Regression we achieved a RMSE of 0.136, scoring significantly better than OLS. This result shows us that our OLS model was likely overfitting. The performance can be further decreased to 0.128 by removing outliers from the dataset.

2.4. Lasso Regression

Lasso Regression is very similar to Ridge Regression, in that it performs L1 regularization instead of L2 regularization. A difference to the Ridge Regression Model is that in the Lasso Regression some coefficients can become zero if the tuning parameter λ is sufficiently large and thus are removed from the model. L2 regularization as mentioned in the Ridge Regression chapter will not set any coefficients to zero.

The goal of Lasso Regression is to minimize the following formula:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

For both Lasso Regression as well as Ridge Regression it is crucial to set a good value for the tuning parameter λ.

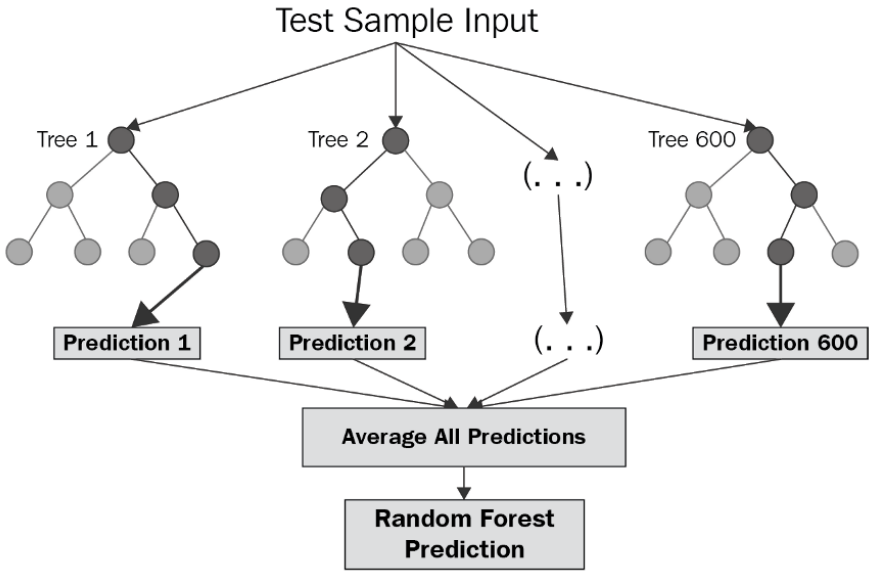
For the Lasso Regression we achieved a RMSE of 0.133. When comparing the RMSE from Lasso Regression with the RMSE of the Ridge Regression, Lasso performs slightly better than Ridge and shows a big improvement compared to the RMSE of OLS. The RMSE can be enhanced further to a RMSE of 0.125 by removing outliers.

2.5. Random Forest

Random Forest is a flexible, easy to use machine learning algorithm that produces great results most of the time with minimal time spent on hyper-parameter tuning. is an ensemble machine learning technique capable of performing both regression and classification tasks using multiple decision trees and a statistical technique called bagging. Bagging along with boosting are two of the most popular ensemble techniques which aim to tackle high variance and high bias. Random Forest utilizes two concepts:

1. Random sampling of training observations when building trees (bootstrapping)
2. Random subsets of features for splitting nodes

Random forest builds multiple decision trees and merge their predictions together to get a more accurate and stable prediction rather than relying on individual decision trees. The fundamental idea behind a random forest is to combine the predictions made by many decision trees into a single model. Individually, predictions made by decision trees may not be accurate but combined, the predictions will be closer to the true value on average. The image below depicts a rough overview of how random forest regression works (Chakure 2019).



Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents overfitting. Nonetheless, this model is prone to overfitting in a lot of cases and thus counts as an “unstable” model.

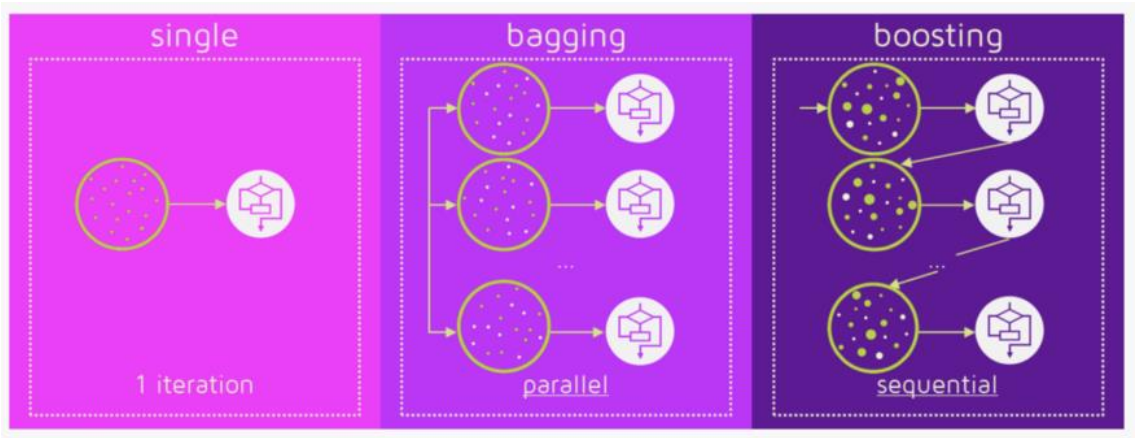
Important parameters used for our model are:

- n_estimators = 1200: number of trees in the forest
- max_depth = 15: depth of the drawn trees
- oob_score = TRUE: whether to use out-of-bag samples to estimate the R-squared on unseen data

The parameters were found through trial and error and allowed us to achieve a RMSE of 0.144. This result is the worst performing model in our analysis after the standard OLS method. Excluding outliers does improve the accuracy, although it still doesn't perform better than the other models without outliers.

2.6. Gradient Boosting

Gradient boosting is a technique attracting attention for its prediction speed and accuracy, especially with large and complex data. Machine learning boosting is a method for creating an ensemble model. It starts by fitting an initial model (e.g. a tree or linear regression) to the data. Then a second model is built that focuses on accurately predicting the cases where the first model performs poorly. The combination of these two models is expected to be better than either model alone. This process, called boosting, is repeated many times iteratively. Each successive model attempts to correct for the shortcomings of the combined boosted ensemble of all previous models. In contrast to the Random Forest Regression which uses the bagging technique that builds independently, this model uses the boosting technique, which builds predictors sequentially. The differences between bagging and boosting are depicted in the graph below (Garrido 2016).



To further give intuition on this model, you can think of the residuals that result from a basic OLS regression. If we consider these residuals as mistakes, we can argue that if we are able to see some pattern of residuals around zero, we can leverage that pattern to fit a model. So, the intuition behind this algorithm is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and thus making it better. Once we reach a stage where residuals do not have any pattern that could be further modeled, we can stop modeling the residuals. With this procedure we are minimizing our loss function, such that test RMSE reaches its minimum. Moreover, we have more parameters to put into this model from the Random Forest model. For one learning rate, which in our case performed well with a value of 0.01, and it shrinks the contribution of each tree by the parameter. We decided to use huber loss as our loss objective as we did not want to have outliers make too big of an impact.

Our analysis shows that the Gradient Boosting model performs best out of the chosen models with a RMSE of 0.120.

2.7. Neural Network

For our most flexible model, we decided to implement an Artificial Neural Network to see if it can reduce the Testing MSE compared to the previous models. While Neural Networks are most commonly used for classification tasks, they can also be used for predictions of continuous values.

In general, a Neural Network can mimic any regression-based model if it is defined correctly. In the simplest model, a Neural Network could have 1 hidden layer with 1 neuron, using a linear activation function and the mean squared error as the loss objective. In essence, this Neural Network would mimic an OLS linear regression model. As we already know that OLS did not perform too well on our dataset (See Chapter 2.2) so we decided to make our Neural Network more complex.

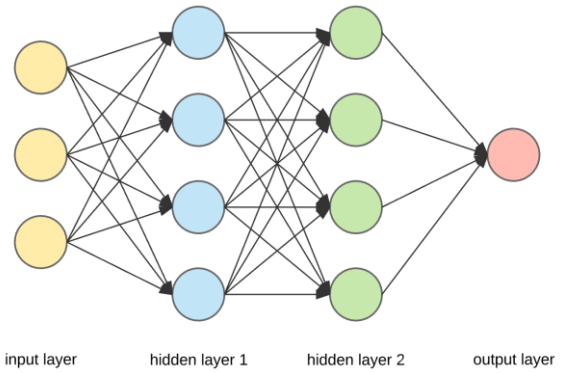
In their simplest form, Artificial Neural Networks have the following types of layers.

1. Input Layer
2. (Multiple) Hidden Layers
3. Output Layer

Where each layer is comprised of different neurons, which in themselves are comprised of a pre-activation and an activation function. (Wang 2003)

Depending on the actual prediction or classification task at hand, one can change and tweak the amount of layers, the amount of neurons within the layers, the type of layers, their activation functions, the loss objective, and how the neurons and layers are connected.

For our rather simple continuous prediction task we decided to implement a sequential model, meaning that every neuron is fully connected to each neuron of the previous layer and the following layer. The graph below depicts how these sequential models are built (Dertat 2017).



In our case, the input layer consists of 255 X values, which are our dependent variables, and 1 bias term. This info is then passed to our 3 hidden layers, which are all Dense layers with 128, 64, and 32 neurons. As we are predicting 1 continuous value, we add 1 final output layer consisting of 1 single neuron. The table below depicts the hidden layers and output layer of our model.

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_24 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_25 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_26 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_27 (Dense)             (None, 1)                 33
=================================================================
Total params: 43,265
Trainable params: 43,265
Non-trainable params: 0
_________________________________________________________________
```

The final layer structure was found through trial and error, where we started from a very simple model with only one hidden layer, and then gradually added layers while carefully watching the validation loss in order to avoid overfitting. Each layer uses the activation function ReLU, which is a commonly used activation function as it allows for fast computation. As our input variables are all positive, the dying ReLU problem did not appear to have an impact on our results. Using other activation functions such as the hyperbolic tangent function did not improve the results, but it did allow for faster convergence when training. This layer structure seems to perform surprisingly well and deviations from it did not achieve any better results.

Our loss objective was the mean absolute error, while we used adaptive moment optimization (Adam). Changing any of these two inputs did not improve our result, however we decided to use a dynamic learning rate for our optimizer. If the validation error did not improve over 5 epochs (each consisting of 37 steps) we reduce the learning rate by a factor of 0.01. This allows us to go even deeper into the minima that the model is converting to. Additionally, to avoid overfitting we stopped training the model when the validation error did not improve after 100 epochs, and then restored the best weights of the model.

The implementation described achieved a RMSE of 0.131. This is significantly better than the results we achieved with OLS, Lasso, Ridge and Random Forest, however, Gradient Boosting is still outperforming our Neural Network. One thing to note is that our model is strongly influenced by Outliers. If we get rid of the outliers (as will be described in Chapter 3), our model's RMSE improves to 0.113, which makes it a significantly better result than that of OLS, Ridge, Lasso and Random Forest. It is important to note that replication of this Neural Network might not always yield the same results. The results are dependent on the initialization weights of the network, the time used to train the model, and the split of testing and training data.

One big disadvantage of the Neural Network is its limited interpretability. We do not understand which variables affect the output, nor do we know how they affect the output. Models like OLS Lasso and Ridge allow for better interpretability as we can look at each single variable and its effect on the price individually.

2.8. Hybrid Predictions

As all of the models above performed fairly well, we were curious to see if we could improve upon the RMSE by mixing many models together. We computed a weighted average of the different predictions, using all models except OLS. The idea behind this approach is that each model is overfitting the data in some way, and that a weighted average of the models will allow models to compensate for the specific overfitting of the other models.

After trying out many different combinations, we achieved a very impressive result when mixing the models with the following weights.

Model	*Weight*
OLS	0.0
Ridge	0.15
Lasso	0.15
Random Forest	0.10
Gradient Boosting	0.35
Neural Network	0.25

This weighting allowed us to reduce the Test RMSE (including outliers) to 0.078. As this is significantly lower than any of the models alone, we can conclude that the hybrid prediction approach worked the best for prediction. Unsurprisingly, choosing a relatively large weight for the best performing models yields the best results. For comparison of other weight distributions and their respective RMSEs, check Table A1 in the Appendix.

3. Results and Conclusion

All of the models described above did reasonably well, however there are some strong differences in the resulting RMSE of the models. The results of each model are shown in the table below.

Model	*Test RMSE (incl. outliers)*	*Test RMSE (excl. outliers) **
OLS	0.184	0.166
Ridge	0.136	0.128
Lasso	0.133	0.125
Random Forest	0.144	0.132
Gradient Boosting	0.120	0.108
Neural Network	0.126	0.113
Hybrid Model**	0.078	0.078

** Test RMS excluding outliers is always lower than the RMSE with outliers, as outliers were dropped in both the training and testing sets.*

*** The Hybrid Model and the respective weights used are described in Chapter 2.8*

As already described in Chapter 2.8, the Hybrid Model outperforms all singular models by a significant margin. This is most likely due to each model overfitting specific parts of the training data. The hybrid model allows the models to compensate for each other's shortcomings.

Additionally, we can compare which models are affected by the outliers and how strongly they are affected. We can see that all single models have a lower RMSE when excluding the outliers, with Gradient Boosting, Neural Network, and Random Forest benefitting most. The Hybrid Model is not significantly affected outliers, further showing that Hybrid Models do not suffer from overfitting as much as the singular models.

Based on these predictions, a number of further works could be done. On one hand, prediction accuracy could be further improved with more detailed feature engineering. This could capture more complex relationships and relevant interactions between predictors. While increasing the number of predictors may produce more accurate predictions in this specific dataset, it will make the results less generalizable. On the other hand, testing our methodology on similar datasets may lead to interesting insights. Finding a model that performs well on various datasets would allow comparisons between different cities which collect data on different variables. Such an approach would allow us to use already available local datasets that contain a lot of information. Compiling larger scale data is often very costly and demands a focus on a limited number of predictors. Avoiding this problem could facilitate investigating the house price differences between different places.

References

Clapp, J., & Giaccotto, C. (2002). Evaluating House Price Forecasts. *Journal of Real Estate Research*, 24(1): 1-26. https://www.tandfonline.com/doi/abs/10.1080/10835547.2002.12091087

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): 1-15. https://doi.org/10.1080/10691898.2011.11889627

Dertat, Arden. (2017) Applied Deep Learning - Part 1: Artificial Neural Networks. *Towards Data Science*. https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning with Applications in R. Springer, New York.

Kahr, J., & Thomsett, M. C. (2005). Real Estate Market Valuation and Analysis. Wiley, Hoboken (NJ).

Limsombunchai, L., 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

Wang SC. (2003). Artificial Neural Network. In: Interdisciplinary Computing in Java Programming. *The Springer International Series in Engineering and Computer Science*, vol 743. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-0377-4_5

Chakure, Afroz. 2019. Random Forest Regression. *Medium.* https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f

Garrido, Ana Porras. 2016. What is the difference between Bagging and Boosting?. *Quantdare - The scientific blog of ETS Asset Management Factory.* https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

Appendix

Table A1: Hybrid Model Performance Based on weighting of Predictions

*Model**	*Ridge*	*Lasso*	*Random Forest*	*Gradient Boosting*	*Neural Network*	*Test RMSE (incl. outliers)*
Model 1	0.15	0.15	0.10	0.30	0.30	0.081
Model 2	0.25	0.15	0.10	0.30	0.30	0.083
Model 3	0.33	0.00	0.00	0.34	0.33	0.082
Model 4	0.15	0.15	0.10	0.35	0.25	0.077

** Table A1 depicts only selected few models, many more were trained but are excluded from this table.*

Table A2: List of explanatory variables

MSSubClass: The building class

MSZoning: The general zoning classification

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access

Alley: Type of alley access

LotShape: General shape of property

LandContour: Flatness of the property

Utilities: Type of utilities available

LotConfig: Lot configuration

LandSlope: Slope of property

Neighborhood: Physical locations within Ames city limits

Condition1: Proximity to main road or railroad

Condition2: Proximity to main road or railroad (if a second is present)

BldgType: Type of dwelling

HouseStyle: Style of dwelling

OverallQual: Overall material and finish quality

OverallCond: Overall condition rating

YearBuilt: Original construction date

YearRemodAdd: Remodel date

RoofStyle: Type of roof

RoofMatl: Roof material

Exterior1st: Exterior covering on house

Exterior2nd: Exterior covering on house (if more than one material)

MasVnrType: Masonry veneer type

MasVnrArea: Masonry veneer area in square feet

ExterQual: Exterior material quality

ExterCond: Present condition of the material on the exterior

Foundation: Type of foundation

BsmtQual: Height of the basement

BsmtCond: General condition of the basement

BsmtExposure: Walkout or garden level basement walls

BsmtFinType1: Quality of basement finished area

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Quality of second finished area (if present)

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

HeatingQC: Heating quality and condition

CentralAir: Central air conditioning

Electrical: Electrical system

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Number of bedrooms above basement level

Kitchen: Number of kitchens

KitchenQual: Kitchen quality

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality rating

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

GarageType: Garage location

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

GarageCond: Garage condition

PavedDrive: Paved driveway

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Fence: Fence quality

MiscFeature: Miscellaneous feature not covered in other categories

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold

YrSold: Year Sold

SaleType: Type of sale

SaleCondition: Condition of sale