# Predicting House Prices: ML Regression Techniques

## Introduction to Dataset:

The Ames Housing Dataset was compiled for a Kaggle data science competition where participants can practice their regression and feature engineering skills. According to the website, “with 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition’s challenge is to predict the final price of each home”[1]. Hopefully, this dataset will prove that a lot more influences the house than the number of bedrooms or whether there is a basement.

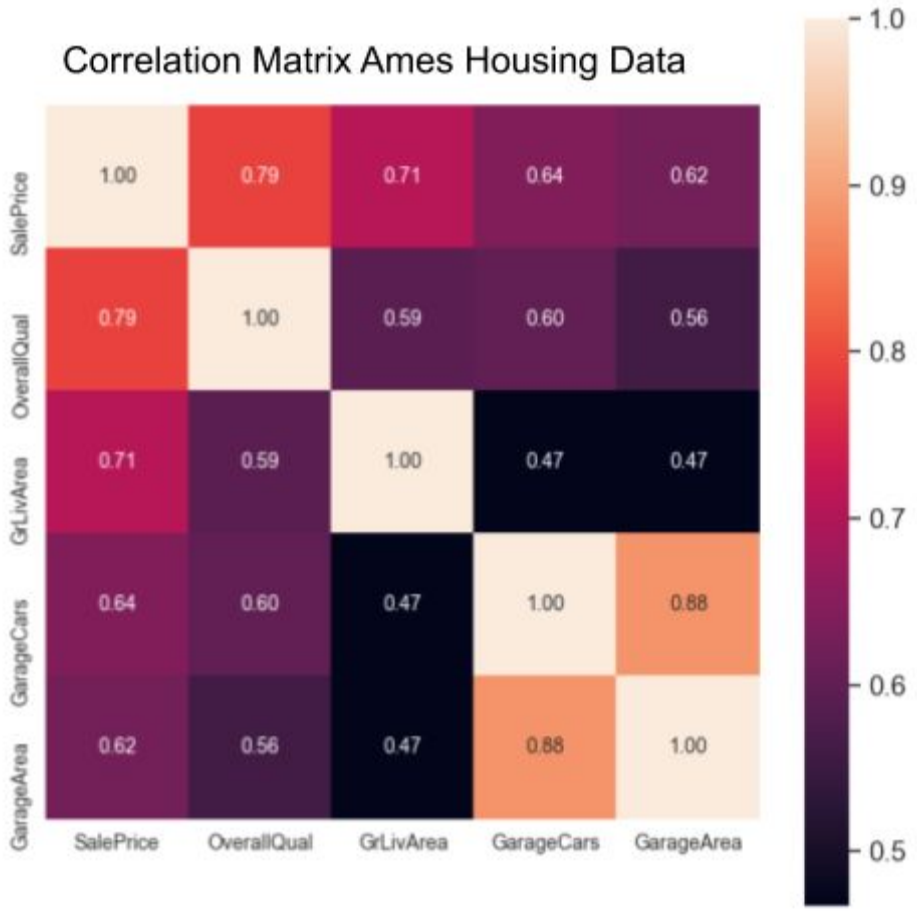
I believe this dataset is interesting because it offers a chance to compare different regression techniques. I believe that the dataset is non-trivial because it consists of over a thousand data points which each have 80 features. Some of these features are numerical, while others are categorical which means that some feature engineering must be performed before the machine learning algorithms can be applied.

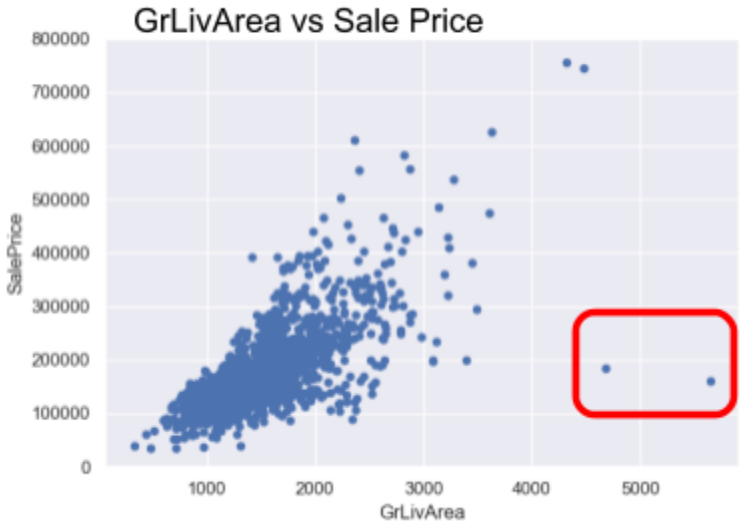
The first task at hand is to gain a basic understanding of the dataset. To do this, a heat map was created to see which features correlate most with the target variable. As can be seen from the heatmap, the feature most correlated with the house’s sale price is “OverallQual”, a categorical variable describing the overall quality of the home. The next most correlated feature is “GrLivArea” which is the above ground living area in square feet. Unlike overall quality, this is a continuous variable so it can be useful to plot it versus the target variable. As can be seen from the plot, there are some clear outliers which can be safely removed to improve the potential performance of the regression algorithms. There are several more steps of data cleanup and normalization to do such as removing columns with missing data and normalizing numerical data. Normalization is important since the data are of vastly different scales.[2]

---

[1] "House Prices: Advanced Regression Techniques | Kaggle." https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data. Accessed 6 Dec. 2019.

[2] "How, When and Why Should You Normalize / Standardize ...." https://medium.com/@swethalakshmanan14/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff. Accessed 6 Dec. 2019.




## Description of Algorithms:

**Kernel Ridge Regression:**
Ridge regression is a form or regression that is good at analyzing data that suffers from multicollinearity. Multicollinearity is defined as the existence of near-linear relationships among independent variables. This kind of scenario is likely to occur in the Ames Housing Dataset because many features such as “GarageCars” (the number of cars that can fit in the garage) and “GarageArea” (the square footage of the garage) correlate strongly with each other (refer to the heatmap on the previous page). Even though it is sometimes possible to

remove strongly correlated independent variables, there are other sources of multicollinearity such as the existence of *outliers* or *physical constraints* in the model. For this reason, ridge regression is a good choice for this dataset.[3]

Ridge regression works similarly to least-squares regression. The ridge coefficients minimize the sum of the squared residuals with a regularization term. The larger the value of the regularization term, the smaller the coefficients are forced to be. This is used to combat overfitting. In the python library scikit-learn, the regularization term is given the name “alpha”. Larger values of alpha will increase the bias of the model (since the ridge coefficients will be more penalized) while smaller values will reduce the bias (and increase the variance) of the model[4]. The algorithm also uses the kernel trick to transform the data before fitting. This allows fitting of non-linear models. There are several different kernels available to select from. These include polynomial kernels of arbitrary degree, sigmoid, radial-basis function, and chi-squared. Two commonly used kernels in regression are the polynomial and rbf kernels and these are compared later on in the report.

**K-Neighbors Regression:**
K-Neighbors regression can be used to predict the value of a new data point based on how closely it resembles the k-nearest points in the training set. The predicted value is calculated based on an interpolation of the target value based on the k-neighbors[5]. As with Kernel Ridge, it is important to standardize the dataset first to account for different measurement scales.

There are a number of parameters to tune but the main two are the number of neighbors and the distance metric used. As with KNN classification, using n=1 neighbors will cause the model to overfit while selecting an n value that is too high will cause underfit. The key is to find a sweet spot between the two. The distance metrics to choose between are the standard euclidean distance, and the manhattan distance[6].

**Neural Network:**
Artificial Neural Networks (ANNs), like their biological counterparts, are comprised of individual elements known as neurons. In our case, the input layer of an ANN needs to consist of the same number of neurons as there are features in the dataset while the output layer will simply have one neuron to predict the house price[7]. A Deep Neural Network (DNN) has more than one hidden layer. In this project, the Keras library will be used to build the neural network. For each layer after the input layer the dimensionality of the output space alone needs to be specified.

Out of all the regression algorithms, the Neural Network has the most hyperparameters to tune. Firstly, for each hidden layer, the dimensionality of the output space needs to be specified. Next, it is important to specify an activation function for the neurons in that layer. In addition, there has to be some method to determine how to initialize the weights and biases of the neurons for each layer, and there are many ways to regularize and/or

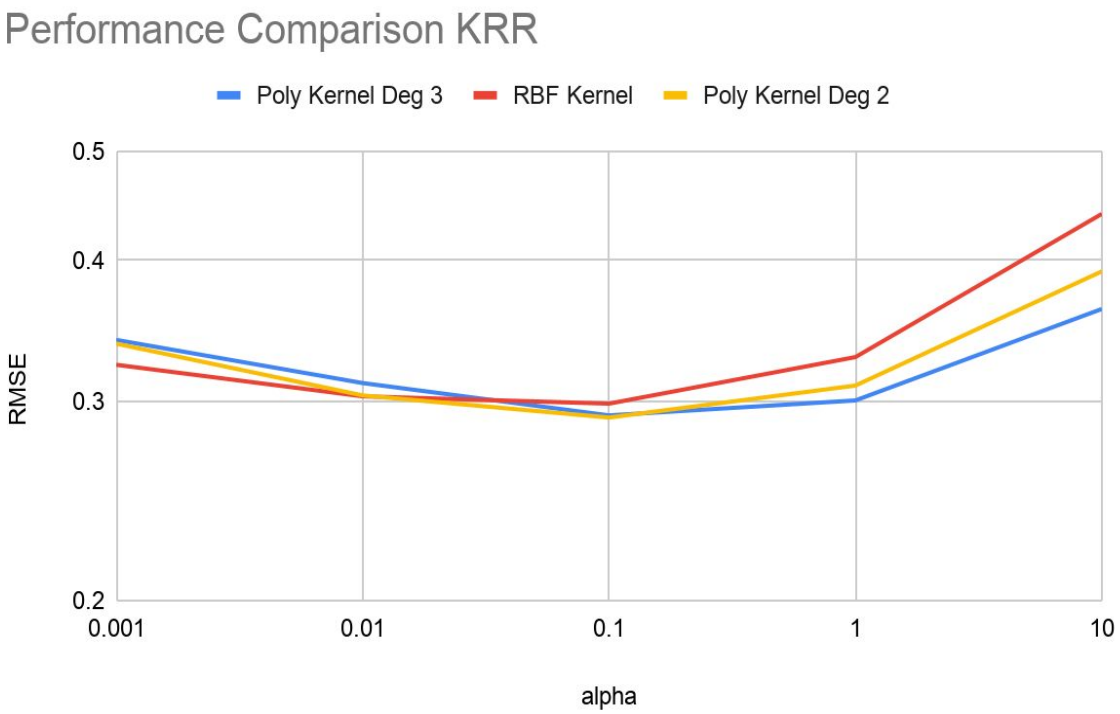
---

[3] "Ridge Regression." https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge_Regression.pdf. Accessed 6 Dec. 2019.
[4] "sklearn.kernel_ridge.KernelRidge — scikit-learn 0.22 ...." http://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html. Accessed 6 Dec. 2019.
[5] "A Practical Introduction to K-Nearest ...." 22 Aug. 2018, https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/. Accessed 6 Dec. 2019.
[6] "sklearn.neighbors.KNeighborsRegressor — scikit-learn 0.22 ...." http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html. Accessed 6 Dec. 2019.
[7] "Complete Guide to Artificial Neural Network Concepts & Models." https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/. Accessed 6 Dec. 2019.

constrain these weights and biases. Lastly, the model needs to be compiled and then fit. There are even more parameters to consider for these steps which will be addressed in the next section.

## Tuning of Hperparameters:

**Ridge Regression:**
For ridge regression, there are two main parameters to select. Alpha, the regularization parameter, and the type of kernel to use. A good way to tune these hyperparameters is to use cross validation. Cross validation involves first dividing the dataset into k groups. Next, the algorithm is fitted to k-1 groups. The “kth” group is held out as the test group and used to evaluate the performance of the algorithm. Usually the k value selected ranges from 5-10[8]. For this assignment, a k-value of 5 was chosen based on the size of the dataset and the time required to train each of the algorithms. Different values for alpha, and different kernels are chosen for each of the training models.

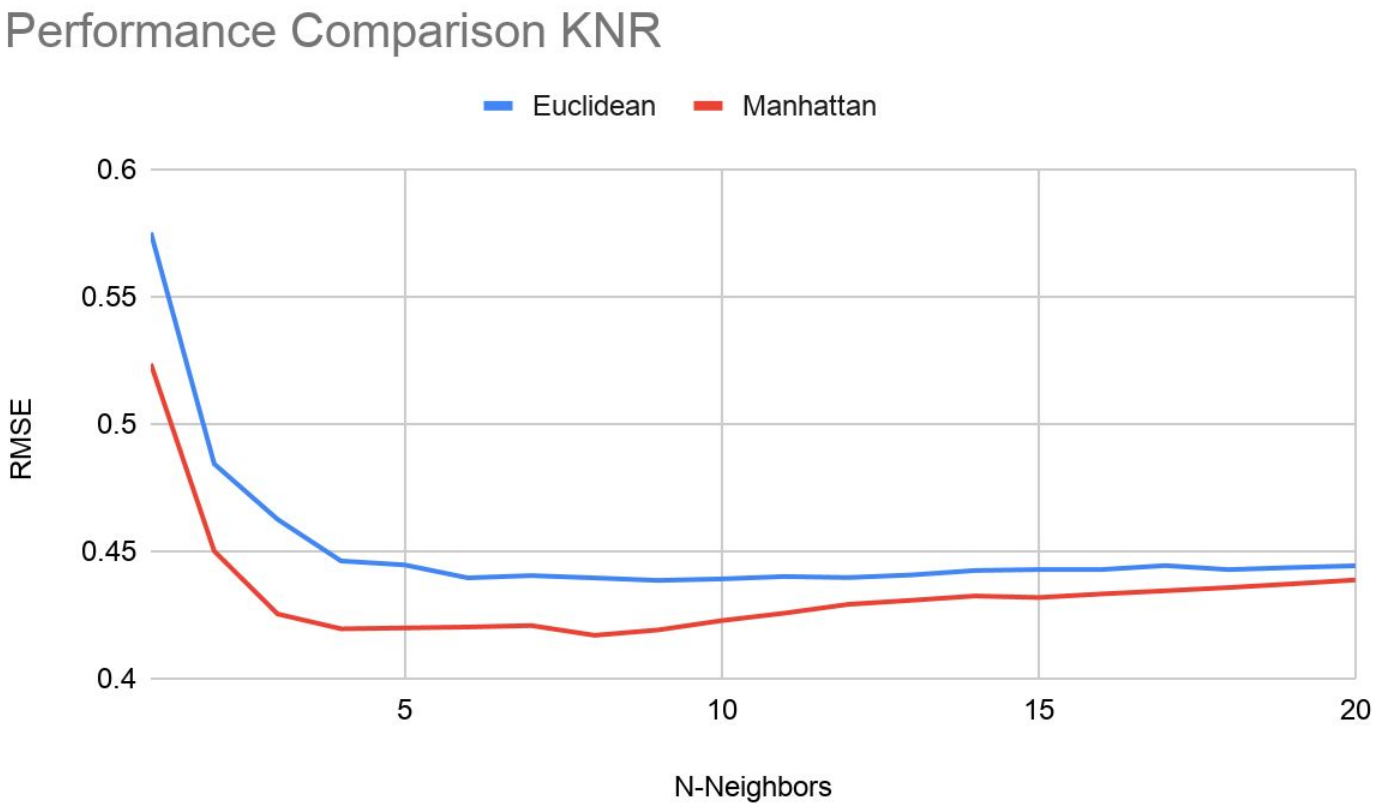



After cross validation, the model with the best score will be selected for evaluation with a held-out testing dataset. The sklearn library offers a function called GridSearchCV which allows one to try cross validation with several different input parameters to find the best fit. From the initial cross validation, it can be assumed that the best parameters will have an alpha value close to 0.1 and likely use a polynomial kernel of degree 3. This means that the scope of the grid search can be narrowed down. The best parameters can subsequently be used to build the model. The result of the grid search was an alpha value of 0.15

[8] "Cross Validation Explained: Evaluating estimator performance.." 26 Nov. 2018, https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85. Accessed 6 Dec. 2019.

**K-Neighbors Regression:**
As with kernel ridge, cross validation was performed to compare the Euclidean and Manhattan distance metrics. As can be seen, manhattan distance has the better performance for all values of k. In addition, it appears that the best value for k is in the range of 5-10.

In order to select the optimal K value, it is possible to run a cross-validated grid search. This will automatically select the optimal number of neighbors to use. We will set the weights of each point to be equal and compare the number of neighbors and the distance metrics. After performing a grid search, it was determined that k=8 was the best parameter for the number of neighbors.




In the table below, the time required to perform the grid search for both algorithms is listed. As can be seen, K-Neighbors Regression had a much longer elapsed time for the grid search since it had to reclassify the entire dataset for each new k value.

| Algorithm for GridSearchCV | Elapsed Time |
|---|---|
| Kernel Ridge Regression | 2.1747 |
| K Neighbors Regression | 8.208 |

**Neural Network:**

For the neural network, the “relu” activation function was selected for the hidden layers. The choice of activation function does not what the neural network can learn - only how long it takes to learn. For this reason, ‘relu’ was selected because it is the most computationally efficient[9]. Only two hidden layers were used since as the number of layers increases, the difficulty of training the NN also increases. In addition, from research it was determined that the number of hidden layers necessary for training a neural network is rarely more than 1 in many applications[10]. Four iterations are tested with hidden layer dimensionalities of 128, 256, 512 and 1024. From this comparison, it appears that the dimensionality of the hidden layers does not have much effect on the loss or the elapsed time so 256 was chosen to be used with the test data.

| Hidden Layer Dimensionality | Loss | Elapsed Time |
|---|---|---|
| 128 | 0.4376 | 0s |
| 256 | 0.3866 | 0s |
| 512 | 0.3546 | 1s |
| 1024 | 0.3786 | 0s |

## Comparing Algorithm Performance:

**Performance Metrics:**
After running the tuned algorithms on the test data, statistical tests can be performed to evaluate their predictive performance. The mean absolute error involves finding every individual difference between the predicted and expected values and then dividing that value by the number of predictions. The RMSE does the same but with squared errors. This means the RMSE is more sensitive to outliers (really bad predictions) than the MAE[11]. For this assignment, the RMSE was used since large errors are particularly undesirable. In addition, the loss functions for the regression algorithms work to minimize the squared errors rather than the absolute errors. It makes sense then to compare the algorithms based on the RMSE rather than the MAE.

The second performance metric used is the $R^2$ Score. The $R^2$ score is calculated by dividing the total variation by the explained variation in the model. It is a score between 0 and 1 with 1 being the best. Generally, the better the model fits the data, the higher the $R^2$ score will be[12]. There are some caveats to the $R^2$ to be aware of. The larger the dataset, the larger the $R^2$ score even if the addition of data does not greatly improve the fit. In addition, a high $R^2$ score could be the result of overfit so it needs to be used in conjunction with the RMSE score and other best practices such as cross validation and using a held-out test dataset for the final performance evaluation.

---

[9] "7 Types of Activation Functions in Neural Networks: How to ...." https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/. Accessed 6 Dec. 2019.
[10] "How to choose the number of hidden layers and nodes in a ...." 20 Dec. 2011, https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw. Accessed 6 Dec. 2019.
[11] "Understand Regression Performance Metrics - Becoming ...." 2 Jan. 2019, https://becominghuman.ai/understand-regression-performance-metrics-bdb0e7fcc1b3. Accessed 6 Dec. 2019.
[12] "Regression Analysis: How Do I Interpret R-squared and Assess." 30 May. 2013, https://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit. Accessed 6 Dec. 2019.

**Performance Comparison:**
Below is a performance comparison of each of the algorithms used. For this comparison, each algorithm was tasked with making predictions against a held-out test dataset. Interpreting the table below, keep in mind that a lower RMSE score and a higher $R^2$ score indicate better performance.

| Algorithm | $R^2$ Score | RMSE Score |
|---|---|---|
| **Kernel Ridge Regression** | 0.925084365675667 | 21,908.970464392896 |
| **K Neighbors Regression** | 0.8331770975912022 | 29,543.98627149522 |
| **Neural Network** | 0.894424445078285 | 23,502.98063905342 |

Evaluating the RMSE score, it may look like the algorithms were extremely inaccurate; however, it is important to remember that the sale price of houses is measured in hundreds of thousands. The mean sale price for a house in the test dataset was $174,622 so the RMSE was actually quite good. With more time for this assignment, more feature engineering and tuning of hyperparameters can yield better results. Looking at the $R^2$ score, it appears that all of the algorithms performed reasonably well with KRR being the clear winner. The $R^2$ is high enough to say that the regression algorithm does a good job of accounting for the variation in the sale price.

Elapsed time is an important metric when evaluating machine learning algorithms. Regardless of performance, time is a scarce resource and there is a trade-off between potential accuracy of the model vs the time it takes to produce results. In this case, once the parameters of the model were selected, the elapsed time to fit each model to the test data and make predictions was trivially short. For this reason, elapsed time was more of a factor with regards to tuning the hyperparameters of each algorithm. As can be seen from the previous section, kernel ridge regression was the fastest overall to tune while the neural network was unsurprisingly the slowest overall.

## Conclusion:

In conclusion, the most effective regression technique turned out to be kernel ridge regression. It had the lowest RMSE score and the highest $R^2$ score. In addition, kernel ridge regression has very few hyperparameters to tune, while other algorithms such as the neural network have several more. The fact that kernel ridge regression worked best was not surprising. As stated in the algorithm description, KRR works well with multicollinear data which is exactly the type of dataset used in this assignment.

Although reasonable performance was obtained during this project. There are several ways in which the performance could have been improved given more time. The first would be more feature engineering. Instead of removing missing data, many of the dropped features can be kept by reading the data description and filling in reasonable values for the missing data. In addition, more information could be extracted by combining, or

adding new features to the dataset. Lastly, there are other regression algorithms that have been known to perform well on this kind of dataset such as elastic net and LASSO regression[13].

**Note:** When running the code, sometimes the values generated will be slightly different from those listed in the report. Despite this, the overall performance of the algorithms stays reasonably constant even if the selected hyperparameters may change very slightly.

[13] "Top Machine Learning Algorithms You Should Know to ...." https://becominghuman.ai/top-machine-learning-algorithms-you-should-know-to-become-a-data-scientist-17b16bc85077. Accessed 6 Dec. 2019.

## Acknowledgements:

**Example Code and Tutorials:**
Below are links to some example code and python notebook tutorials that were very helpful with data cleaning. These articles taught me that it is important to get an understanding of the dataset first instead of jumping in and applying machine learning straight away. Standardizing and normalizing data is important when each feature is measured on a completely different scale. In addition, viewing correlations before performing regression can help a data scientist know what to expect.

www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python#1.-So...-What-can-we-expect?
https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard/data#Data-Processing
https://www.kaggle.com/jsvishnuj/data-cleaning-and-k-nearest-neighbors-algorithm

This article below is a tutorial for creating a neural network for regression in Keras. Since this assignment was my first time using Keras this article was extremely valuable.

https://towardsdatascience.com/deep-neural-networks-for-regression-problems-81321897ca33

**Software Libraries Used:**
Pandas
https://pandas.pydata.org/

Numpy
https://numpy.org/

SciKitLearn
https://scikit-learn.org/stable/

Keras
https://keras.io/

Matplotlib
https://matplotlib.org/