Final Report - House Prices Prediction

Kaijun Zhang
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
kzhang29@ncsu.edu

Wesley Meredith
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
wmeredi@ncsu.edu

Yiran Zhu
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
yzhu59@ncsu.edu

Abstract

Accurate prediction of housing prices is crucial for many real estate stakeholders. In this study, we address the task of accurate housing price prediction, specifically focusing on the Ames housing dataset [1]. Utilizing a diverse feature set encompassing both numerical and categorical data, we employ advanced pre-processing techniques to explore feature relationships. We comprehensively compare the predictive performance of four ML models—decision tree, random forest, XGBoost, and artificial neural networks—analyzing their effectiveness in forecasting home values. Furthermore, we investigate the relative impact of feature engineering versus hyper-parameter tuning on our prediction outcomes. A rigorous evaluation on a dedicated test set utilizing RMSE significantly contributes to our comprehensive analysis of housing price prediction.

1 Background

1.1 Problem Description

The central challenge addressed in this project revolves around precise housing price prediction, a critical aspect of the real estate industry. Accurate predictions significantly influence decision-making for various stakeholders in the real estate market. While simpler machine learning models, such as decision trees, can be effective in predicting home values, advanced methods like random forest and XGBoost have also demonstrated success in housing price predictions across diverse regions and contexts [2]. Despite the effectiveness of individual ML models like decision trees, there is a growing interest in exploring ensemble learning techniques to enhance prediction accuracy by aggregating insights from multiple base learners to form a more complex model [3].

The specific focus of this project involves the comparative analysis of four distinct machine learning models: decision tree, XGBoost, artificial neural network (ANN), and random forest. Drawing on the rich literature in ML applications, the project aims to investigate and quantify the impact of two key factors on model performance: feature engineering and hyperparameter tuning. The study delves into whether enhancing the quality of input features or optimizing model parameters holds more significance in improving the accuracy of housing price predictions. The dataset under consideration encompasses a diverse set of numerical and categorical features such as the number of bedrooms, lot size, roof type, and year built, contributing to the complexity of the predictive modeling task.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Through this comprehensive exploration, the project seeks to offer insights into the comparative effectiveness of these machine learning approaches for housing price prediction. It also aims to provide guidance on the emphasis between feature engineering and hyperparameter tuning in optimizing model performance, ultimately contributing to the landscape of ML applications in the real estate sector.

1.2 Literature Survey

The Ames Housing dataset, with its diverse features, non-linear relationships, and outliers, has driven many researchers to explore machine learning models for predictive modeling.

Decision trees have been valued for their simplicity and interpretability, and have been extensively applied in Ames Housing, capturing non-linear relationships and revealing feature importance [5]. Their adaptability to mixed feature types makes decision trees relevant to Ames Housing challenges.

Random forest, an ensemble method based on decision trees, has shown to excel in handling noise and outliers [5]. In Ames Housing, where outliers impact prices, random forest's robustness proves valuable, mitigating overfitting and enhancing generalization.

XGBoost, a gradient-boosted decision tree implementation, has shown to excel in handling complex datasets, capturing intricate patterns for non-linear predictions [3]. Ensemble learning, as seen in XGBoost, enhances performance by aggregating base learners, which would prove to be beneficial in this dataset [4].

Artificial neural networks (ANNs) have proved to be excellent at capturing intricate patterns in a dataset. ANNs show adaptability to diverse complexities in a dataset including non-linear variables, making them a fitting choice for predicting house prices [6].

In conclusion, the literature emphasizes the effectiveness of XGBoost, decision trees, random forest, and artificial neural networks for addressing Ames Housing challenges, offering diverse strengths for insightful and accurate housing price predictions.

2 Proposed Method

2.1 Intuition/Rationale

Data preprocessing is vital for ensuring quality data for machine learning models. It involves addressing missing values, encoding categorical variables, handling outliers for improved accuracy, reducing dimensionality via techniques like PCA to prevent overfitting, performing feature selection to optimize model learning, and engaging in feature engineering to uncover hidden patterns. These steps collectively enhance the dataset's suitability, enabling models to learn effectively and make accurate predictions.

The selection of four distinct machine learning models - decision tree, random forest, XGBoost, and artificial neural networks (ANN) - stems from our pursuit of a comprehensive evaluation across varied modeling paradigms.

The decision to evaluate decision tree and random forest models alongside XGBoost for housing price prediction in the Ames dataset stems from their respective strengths and historical success in handling complex data relationships. Decision trees are chosen for their simplicity and interpretability, effectively capturing non-linear patterns in the dataset. Random forest, an ensemble method based on decision trees, excels in mitigating overfitting and handling noise, which is particularly relevant in the Ames housing dataset known for its diverse features and outliers.

However, the choice of XGBoost as the optimal model was due to its unique ability to handle intricate patterns in complex datasets, its efficiency in processing diverse features, and automated preprocessing capabilities that streamline data handling. Despite the potential of ANNs in capturing patterns, their computational requirements led us to prioritize XGBoost's superior performance and interpretability for this housing price prediction task. This comprehensive approach, leveraging diverse methodologies, positions our model for robust and accurate housing price predictions.

2.2 Algorithm Description

2.2.1 Decision Tree

A decision tree regressor predicts values by splitting data into branches based on feature values, forming a tree structure with decisions at each node. It chooses splits to minimize variance, using Mean Squared Error in our housing prices context. For Ames's housing prices, it analyzes house features, like size and bedrooms, to make predictions, starting from a broad feature like 'Total Living Area' and refining through more detailed ones, offering an interpretable and straightforward model to identify dataset limitations. This method is a simple and easy-to-understand way to test out our dataset and find out its limitations.

2.2.2 Random Forest

A random forest regressor, an ensemble approach for regression, builds multiple decision trees during training and averages their predictions to enhance accuracy and stability. It overcomes decision tree limitations by using bagging—sampling data with replacement—and selecting the best split from a random subset of features, reducing overfitting. In housing price prediction, it uses diverse data samples, like square footage and location, to predict prices with a range of decision trees, capturing complex patterns and stabilizing predictions through averaging.

2.2.3 XGBoost

XGboost differs from other decision tree ensemble methods due to the gradient boosting aspect. It will boost the ensemble learner, but it will apply a gradient descent method to optimize the predictions, and it does this sequentially for each learner by learning the mistakes from the previous ones. XGBoost is an optimal choice for housing price prediction on the Ames dataset due to its efficiency, scalability, and high predictive performance. XGBoost can handle missing values and encode categorical data, eliminating the need for those steps to be done manually and ensures robust predictions. Its regularization techniques prevent can also help prevent overfitting [4].

2.2.4 ANN

We implemented artificial neural networks (ANNs) for predictive modeling, inspired by the human brain's neural structure. The ANN comprises interconnected layers of neurons (Figure 1).

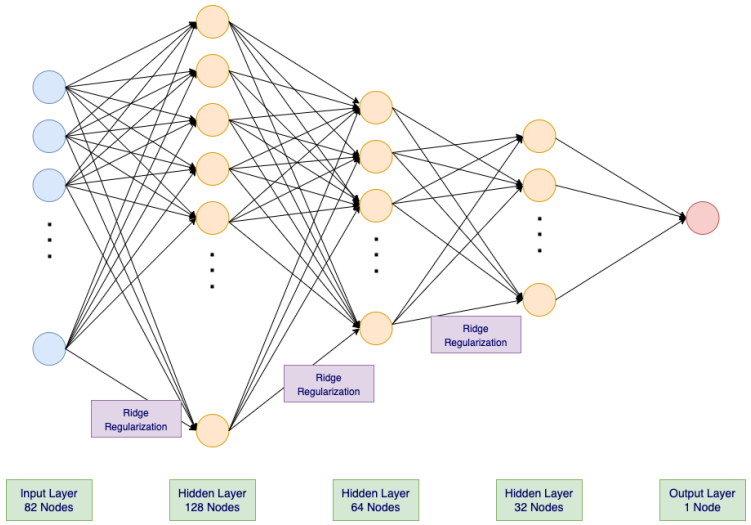


Figure 1: ANN Structure

The ANN structure includes:

- **Input Layer:** Featuring 82 neurons representing house-related dataset features.

- **Hidden Layers:** Three subsequent layers include 128, 64, and 32 neurons, respectively, using the Rectified Linear Activation Function (ReLU) for introducing non-linearity and efficiency in learning complex data relationships.
- **Output Layer:** Tailored for continuous house price prediction using RMSE as the chosen loss function (Equation (1)). This layer's single neuron forecasts house prices in a regression setting.

$$\text{loss} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \tag{1}$$

where n represents the size of the dataset, $\hat{y}_i$ signifies the predicted sale price, and y_i stands for the actual sale price.

We integrated L2 Regularization within the hidden layers to control complexity and prevent overfitting. This regularization technique aimed to optimize the model's performance on limited training data by minimizing a modified cost function involving a penalty parameter λ as shown in Equation (2)

$$\text{cost} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} + \lambda\sum_{j=1}^{p} w_j^2. \tag{2}$$

Here, n represents the size of the dataset, $\hat{y}_i$ signifies the predicted sale price, and y_i stands for the actual sale price, w_js represent the weights of the model, and $\lambda \in [0, +\infty)$ serves as the penalty parameter, determined through 5-fold cross-validation.

By treating the weights w_j as a penalty, our objective was to optimize the model, reducing its sensitivity to the training data and thereby enhancing its performance on unseen data.

3 Plan & Experiment

3.1 Hypothesis

Our main hypothesis for this project is that targeted feature engineering will lead to a more substantial improvement in the predictive performance of machine learning models for housing price prediction on the Ames dataset compared to extensive hyperparameter tuning. We expect that enhancing and creating relevant features based on the dataset's diverse characteristics will play a more pivotal role in increasing predictive accuracy than fine-tuning model parameters. The investigation seeks to determine whether optimizing input features has a more pronounced effect than adjusting model parameters in refining the performance of predictive models.

3.2 Dataset Description

The Ames Housing Dataset comprises a training set consisting of 1460 rows and a testing set containing 1459 rows. Each dataset, apart from the `Id` column, comprises 79 property attributes. The training set includes a column indicating the sale price (`SalePrice`) for each property, while the testing data lacks this column, as its purpose is to enable model builders to predict prices.

These attributes are categorized into two types: numerical data and categorical data. Numerical data encompass features such as lot size (`LotArea`, `LotFrontage`), garage size (`GarageArea`), living area (`GrLivArea`), overall quality (`OverallQual`), year built (`YearBuilt`), and more. On the other hand, categorical data includes descriptors like the type of road/alley access to the property (`Street & Alley`), neighborhood (`Neighborhood`), type of foundation (`Foundation`), pool quality (`PoolQC`), among others.

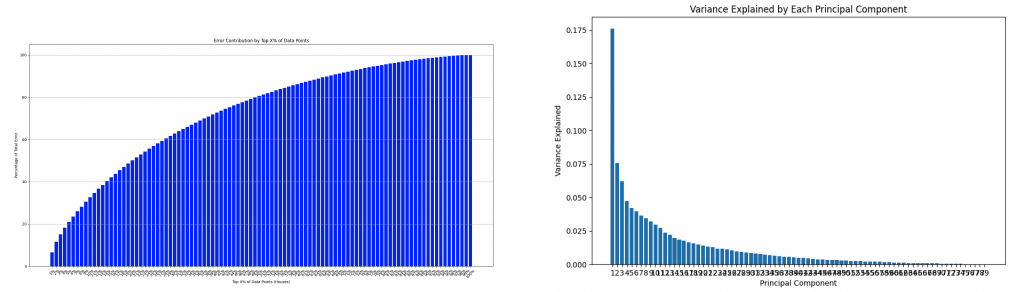
This blend of numerical and categorical features constitutes a diverse array capturing various aspects of residential properties, aiding in the prediction of house sale prices in the Ames area.

Within the datasets, there are `NaN` values. Some of these indicate missing values, while others signify the absence of certain property features. For instance, a `NaN` in the `PoolQC` column implies that the property lacks a pool.

3.3 Data Preprocessing

The initial phase of data preprocessing for the Ames Housing dataset focused on enhancing data quality to build a robust predictive model. We tackled missing values by imputing 'None' for missing categorical data and using mean or median for numerical gaps. Categorical variables were label-encoded to convert categories into numerical values, suitable for model processing. One-Hot-Encoding was initially considered but later dropped due to its limitations with new data in testing. Additionally, the dataset was meticulously reviewed for typos and inaccuracies to preserve data authenticity and ensure accurate model outcomes. Further processing includes:

- **Outlier Identification and Treatment:** We performed an error distribution analysis to identify outliers (Figure 2 (a)), particularly focusing on their impact on the prediction of housing prices.



(a) Error Contribution by Top X percent of data points (b) Variance Explained by Each PC

Figure 2: Outlier Identification & PCA

- **Dimensionality Reduction Consideration:** Principal Component Analysis (PCA) was used to reduce the dimensionality of the dataset, which could lead to simpler models and less overfitting (Figure 2 (b)).
- **Feature Selection:** Removal based on the amount of variance intra-feature was also considered since there were features like utilities or condition2 that consisted of 99% of one class, which intuitively, could be removed since it provides little evidence for the target variable.
- **Feature Engineering** Feature engineering was employed in an effort to clearly define some of the underlying trends/relationships that exist within our dataset, which could boost the model's predictive ability.

3.4 Model Building and Training

3.4.1 Decision Tree & Random Forest

In line with our primary hypothesis focusing on the Ames dataset's housing price prediction, we conducted experiments to scrutinize the impact of various preprocessing techniques on decision tree and random forest models. Our experiments encompassed PCA, feature selection, outlier removal, and feature engineering, each accompanied by a 5-fold Cross-Validation (CV) training process. For both decision tree and random forest models, hyperparameter searches were conducted corresponding to each preprocessing technique used.

3.4.2 XGBoost

See **XGBoost_final.py** on github for relevant code. The training set **train4.csv** has been prepared by removing specified features, applies feature engineering including imputation and label encoding, standardizes the feature set, and trains an XGBoost Regressor. 5-fold CV was also used here. GridSearchCV, which was commented out at the bottom of the code file, was done on a external computing cluster due to the computation demands. The model is then used to predict SalePrice for the test data, and RMSE is calculated for evaluation.

3.4.3 ANN

After completing data preprocessing and feature engineering, we obtained a training dataset that includes the `SalePrice` column and a separate testing dataset without this column. To facilitate model evaluation during training, we randomly set aside 20% of the training set for validation using the `train_test_split` function from the `sklearn` library.

To ensure consistent feature scales and distribution, we utilized `MinMaxScaler` from `sklearn`. This normalization technique adjusts feature values to a specific range, ensuring uniformity across the dataset by fitting and transforming the training data, and applying the same scaling parameters to the validation and test sets.

The ANN model architecture was built using `Keras`. Prior to finalizing the model architecture, diverse experiments were conducted to optimize its structure and performance. Variations in hidden layers (2 or 3), neurons per layer (64/32 or 128/64/32), activation functions (ReLU, Tanh, or Sigmoid), and inclusion of regularization techniques were explored for model optimization.

The model was compiled using the `adam` optimizer and a custom RMSE function as the loss metric for 400 epochs with a batch size of 32. Post-training, the model's performance was assessed on the validation set through RMSE computation. We also visualized the training and validation loss trends using `matplotlib` to observe the model's convergence and potential overfitting tendencies.

Finally, we apply the model to the dataset **test.csv**, and submit the predicted sale prices of the test set to Kaggle, and evaluate the performance of the model on unseen data by Kaggle score[1].

4 Results

4.1 Training Results

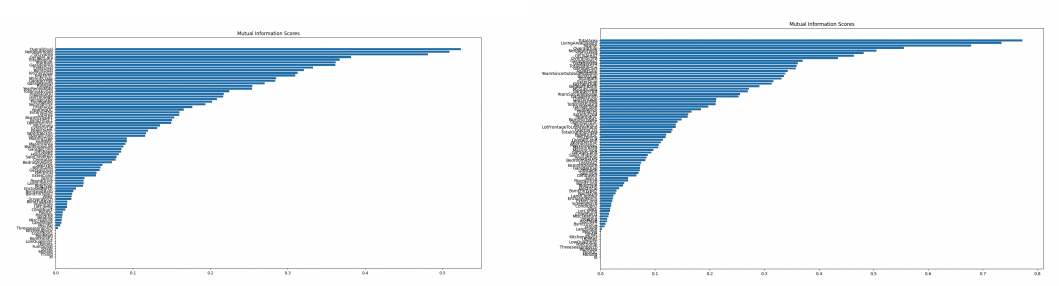
4.1.1 Decision Tree, Random Forest, XGBoost Training Results

For decision tree based models (decision tree, random forest, XGBoost), there were several surprising outcomes while training the model on the four different types of pre-processing methods (refer to Table 1 for results).

- **PCA:** PCA resulted in poorer performance compared to the baseline model. The issue stemmed from PCA's linear nature, which conflicted with the decision tree like model's strength in capturing non-linear relationships. Additionally, PCA's transformed features reduced the model's interpretability, crucial for decision trees like models that thrive on explicit data thresholds and categories.
- **Outlier Removal:** Eliminating top 5 percent of data points contributing most to prediction errors, based on error distribution analysis, decreased the training RMSE, but led to an increased the testing error, which indicates that the outliers are important points to generalize for.
- **Feature Selection:** There was not a significant change in training error or in testing error after doing feature selection on the original dataset, in fact, the test results was slightly worse. This is because while the features removed (variance < 0.03 percent) seemed irrelevant, some features with low variance like havePool has a high correlation with the Sale Price.
- **Feature engineering:** This significantly boosted our predictive models by clearly defining the data's complex relationships. Polynomial features like the square and cube of square footage and room numbers were introduced to grasp non-linear price influences. Interaction features considered the combined effects of variables such as a house's age and size, while ratios like living area to lot size provided deeper spatial understanding. Insights from broader real estate knowledge informed the process, ensuring relevance and depth. We employed mutual information scores to gauge each new feature's value, favoring those that most reduced uncertainty about housing prices. In our experiment, the average Mutual Information score was 0.126 (Figure 3 (a)) before feature engineering, and 0.162 (Figure 3 (b)) after feature engineering, which drastically increased our model's prediction abilities.

[1] Kaggle score measures the average error percentage on unseen test data. Lower is better.

- **Hyper Parameter Tuning:** Hyperparameter grid search did not enhance our model's accuracy, likely because the model was already optimized with existing features, or the search space did not encompass the most impactful hyperparameter combinations. It's also possible that dataset complexity limits further accuracy gains, regardless of parameter adjustments.



(a) Pre-Feature Engineering, average = 0.126 (b) Post-Feature Engineering, average = 0.162

Figure 3: Mutual Information Score

Table 1: Comparison of Techniques

Technique	Decision Tree		Random Forest		XGBoost		ANN	
Baseline	38,187	0.202	33,314	0.178	31,314	0.169	30,542	0.175
PCA	43,873	0.251	36,924	0.198	34,912	0.195	32,483	0.166
Feature Selection	37,876	0.198	33,523	0.175	31,100	0.167	29,302	0.167
Outlier Removal	22,591	0.210	19,501	0.183	18,443	0.173	18,202	0.177
Feature Engineering	32,542	0.168	27,590	0.142	26,598	0.139	21,776	0.137
Hyper-Parameter Tuning	37,910	0.199	32,182	0.176	30,032	0.167	-	

** Note: Values on the left of | represent RMSE, and values on the right represent Kaggle Score (Test result).*

4.1.2 ANN Training Result

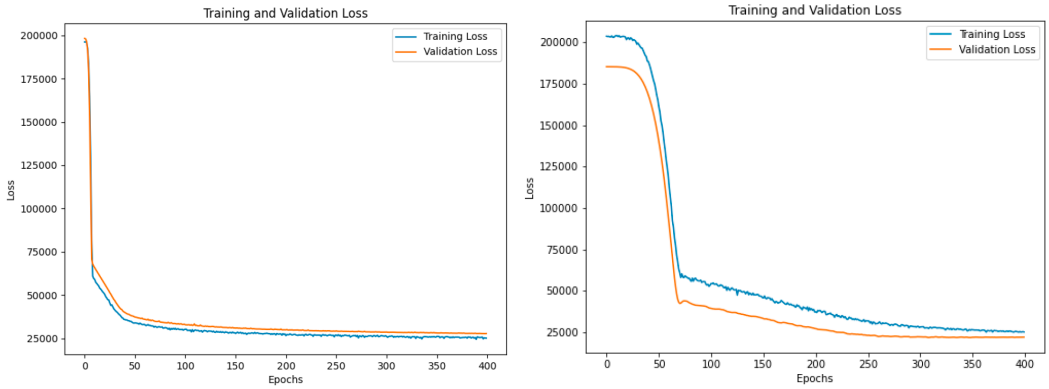
Increasing the hidden layer from two layers (64/32) to three layers (128/64/32) notably reduced the RMSE from a level around 50,000 to approximately 20,000, assuming other variables remained constant. Variations in activation functions showed minimal impact on the outcome; thus, we opted for ReLU due to its simplicity in the final model.

The integration of L2 Regularization significantly enhanced the model's performance. Figure 4 (a) and (b) exhibit the learning curves before and after implementing L2 Regularization. There was a marked improvement in RMSE, decreasing from 27,294 to 21,776, along with an enhancement in the Kaggle score from 0.1872 to 0.13651.

4.2 Model Comparison & Evaluation

The comparison in Table 1 demonstrates that feature engineering notably improved model performance by defining intricate relationships and boosting predictive abilities. Moreover, integrating L2 Regularization into the artificial neural network (ANN) architecture significantly enhanced its effectiveness. Both XGBoost and ANN showcased superior performance in predicting housing prices.

The experimental results largely aligned with our hypothesis, indicating that feature engineering indeed played a pivotal role in enhancing predictive accuracy across models. The emphasis on creating and optimizing relevant features based on the dataset's diverse characteristics demonstrated a more pronounced effect than fine-tuning model parameters. Despite the success in enhancing model performance through feature engineering, the experiment highlighted limitations in hyperparameter tuning, suggesting that further adjustments may be constrained by the dataset's complexity.



(a) Before Using L2 Regularization (b) After Using L2 Regularization

Figure 4: Learning Curves

5 Conclusion

5.1 Lessons Learned

This housing price prediction project using the Ames dataset highlights key lessons for future efforts. Notably, dedicated feature engineering had a greater impact on improving model performance than hyperparameter tuning, thus emphasizing the importance of data preprocessing. The regularization techniques applied to the ANN were highly effective, leading to the lowest prediction error among tested models. This underscores the potential of properly regularized neural networks for the task. PCA did not work well for this dataset due to the abundance of non-linear features.

5.2 Future Work

Future work on the housing price prediction model will look into using K-nearest neighbor (KNN) algorithms, leveraging their similarity-based approach akin to methods on real estate websites, where the value of a home is inferred from closely matching properties. This method could make the model's estimations more understandable and directly comparable for users. Additionally, there's room to expand feature engineering beyond current machine learning literature to include a broader spectrum of real estate knowledge. Exploring a wider range of sources could yield new features that capture more aspects of housing valuation. Finally, experimenting with a mix of brute-force and heuristic methods to generate feature combinations, assessed by their Mutual Information scores, may reveal new, significant predictors that enhance the model's precision.

References

[1] 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.

[2] Ling Zhang. (2023). Housing Price Prediction Using Machine Learning Algorithm. *Journal of World Economy*, 2(3), 18–26.

[3] Shahhosseini, M., Hu, G., Pham, H. (2020). Optimizing Ensemble Weights for Machine Learning Models: A Case Study for Housing Price Prediction. In: Yang, H., Qiu, R., Chen, W. (eds) Smart Service Systems, Operations Management, and Analytics. INFORMS-CSS 2019. *Springer Proceedings in Business and Economics*. Springer, Cham.

[4] Tianqi Chen and Carlos Guestrin. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). *Association for Computing Machinery*, New York, NY, USA, 785–794.

[5] Breiman, L. (2001). Random Forests. *Machine Learning* 45, 5–32.

[6] N. Zulkifley, S. Rahman, U. Nor Hasbiah, I. Ibrahim. (2020). House Price Prediction using a Machine Learning Model: A Survey of Literature. *International Journal of Modern Education and Computer Science*, 12, 46-54.

Appendix

Work Division

- **Model Division**: Kai built decision tree and random forest, Wesley did XGBoost, and Yiran did ANN.
- **Data Cleaning:** For each model, all group members contributed equally to data-cleaning and preparation for each of their respective model.
- **Methods Development:** All group members contributed to this workflow equally.
- **Exploration:** Each group member had their own exploration workflow, but all followed the same procedure.
- **Results Analysis:** Each group member had to analyze their respective model they built.
- **Conclusion Drawing:** All group members came to the same conclusion in a group meeting.
- **Presentation Preparation:** All group members were present at the meetings organized outside of classroom time, and person contributed equally to their respective section.
- **Final Report Creation:** Work was divided equally into sections, for example, Wesley took background and literature survey while Kai did pre-processing/feature engineering, and Yiran doing experiment setup and evaluation.