Name: Ngoc Nguyen
Course: STA467 A
Instructor: Lee Donghyung

FINAL PROJECT REPORT

I. INTRODUCTION

Housing prediction is one of the most classical regression problem in machine learning world. In this project, I want to challenge myself with the Ames Housing dataset from Kaggle which was a more modernized and expanded version of the often cited Boston Housing dataset.

So far, the valuation of residential properties extends far beyond conventional metrics such as bedroom count or aesthetic features like white picket fences or the proximity to an east-west railroad. With an extensive collection of 79 explanatory variables documenting comprehensive aspects of housing properties in this dataset, housing price negotiations are shown to be influenced by numerous less obvious factors that might escape immediate attention—such as basement ceiling height or number of bathrooms.

The primary objective of this modeling exercise is to accurately predict the final sale price of each residence based on these diverse property characteristics, developing a robust predictive framework that captures both obvious and subtle pricing determinants. Ultimately, this study can help house buyers/renters and investors have more insights into the real estate scene.

II. METHODOLOGY

2.1 Data

This dataset comes from a Kaggle competition - called House Prices - Advanced Regression Techniques. Our target variable is `SalePrice`. It contains 1,460 observations of 79 explanatory variables (excluding id and saleprice columns) which describe (almost) every aspect of residential homes in Ames, Iowa. Every observation is unique. There is no more additional information about collecting the data.

To get in-depth information about the competition description and data structure, please refer to this link: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview

2.1.1 Data Cleaning

This section includes the following responsibilities:

- Data inspection (dropping unnecessary variables, check for outliers, duplicate values).
- Handling missing values (using most frequent values, data descriptions or imputation)
- Detailed variable investigation (partitioning into Categorical `cat_columns_df` and Numeric `num_columns_df` to process each).
- In each dataframes, I examine distribution and patterns, test if there is adequate variance, that is class balance inside each feature, find the reason for keeping, otherwise remove that feature.)

- Label encoding for the `cat_columns_df` using one-hot encoding and label encoding, because machine learning models like regression and tree-based require strict numeric input.
- Feature selection/Dimension reduction by feature engineering (those features in `num_columns_df`) and using correlation matrix to eliminate highly correlated variables.
(Very) Detailed implementation can be found in .Rmd file.

As soon as both the categorical and numeric variables were cleanly processed. I concat them back as final training set. Following is a snapshot of data used for training:

	LotShape <dbl>	ExterQual <dbl>	ExterCond <dbl>	BsmtQual <dbl>	BsmtCond <dbl>	BsmtExposure <dbl>	BsmtFinType1 <dbl>	BsmtFinType2 <dbl>	HeatingQC <dbl>
1	3	3	2	3	2	2	6	1	4
2	3	2	2	3	2	4	5	1	4
3	2	3	2	3	2	2	6	1	4
4	2	2	2	2	3	2	5	1	3
5	2	3	2	3	2	3	6	1	4
6	2	2	2	3	2	2	6	1	4

6 rows | 1–10 of 194 columns

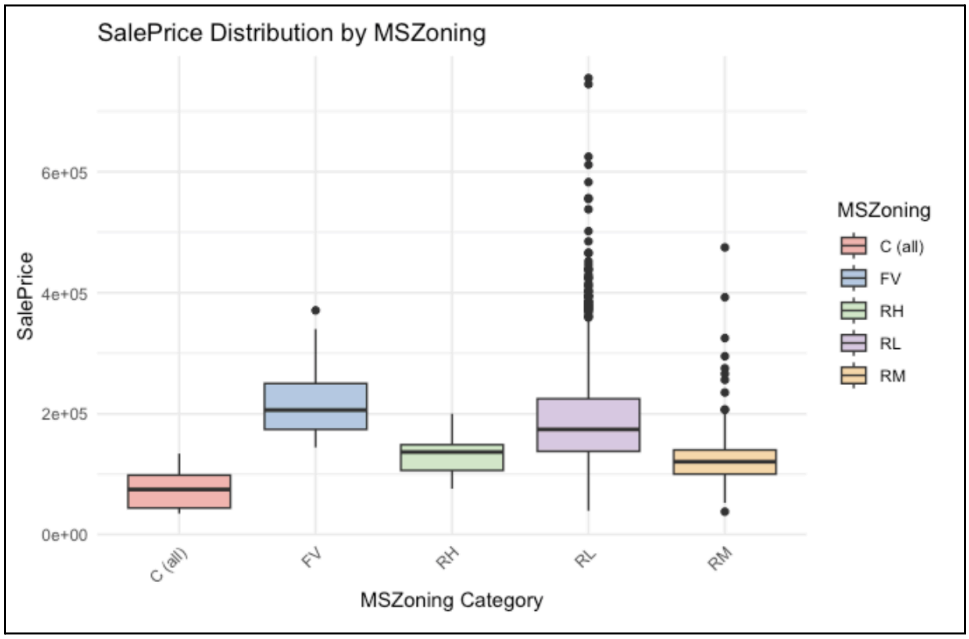
2.2 Software: R-Studio

III. RESULTS

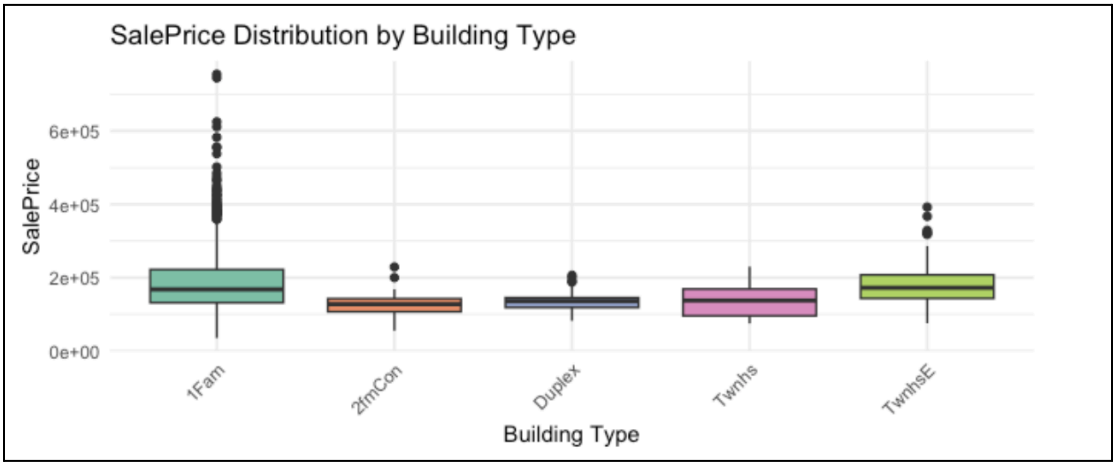
3.1 Exploratory Data Analysis

3.1.1 Plotting Categorical Variables

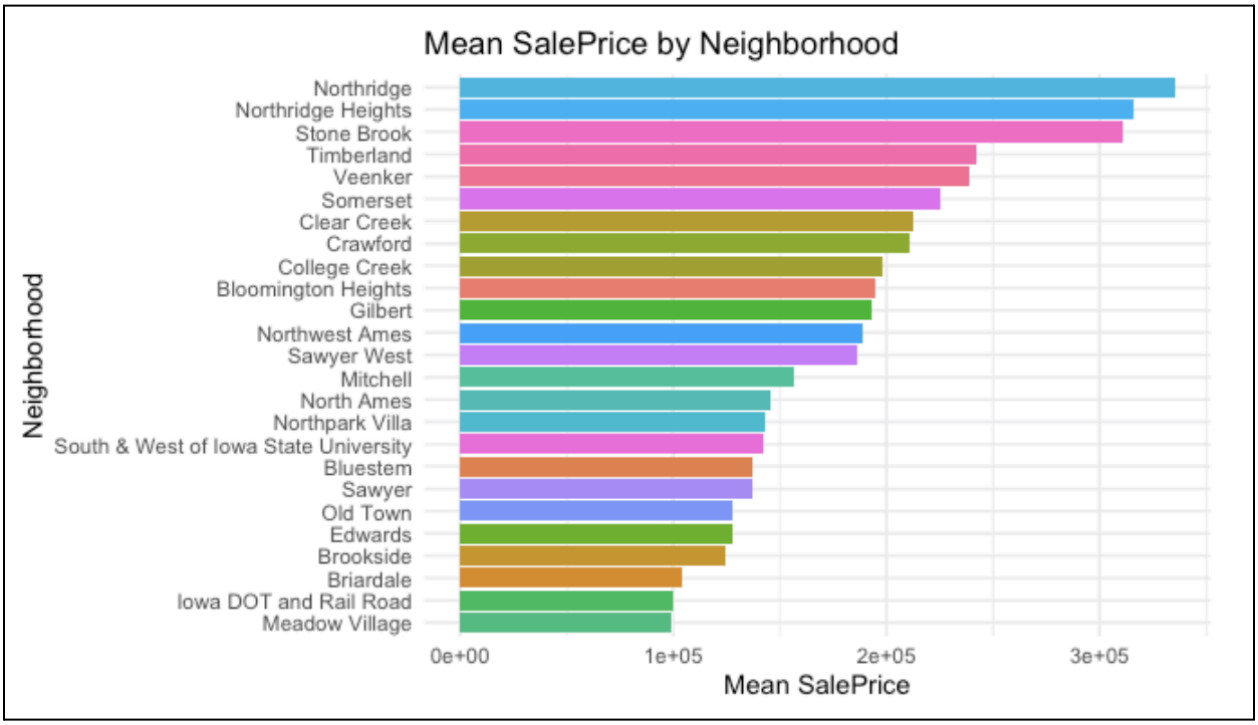
Since there are so many variables (170 categorical variables), I will only do some exploration into some determining factors that people usually based on when making house-buying decisions. First is the general zoning of the house sale, that is, how the land where the house is built is legally designated to be used, for example: for farming purpose, for commercial purpose, residential purpose, etc.



The above plot FV(Floating Village Residential) has overall higher sale price than the other types of land, it is also less variable. In contrast, RL (residential low density) land is highly variable and has lots of outliers. This is important to keep in mind if later we want to improve model performance (by dropping outliers that inflate the errors).
We also want to look at the distribution of building/dwelling type:



It's clear that 1Fam(single-family detached) type has the most and highest housing sales and they are very right-skewed. Next, we want to see people references on the places/venues that are easily accessible from their homes.



From the plot, Northride, Northride Heights, and Stone Brook are the top 3 Ames locations that people want to live close to.

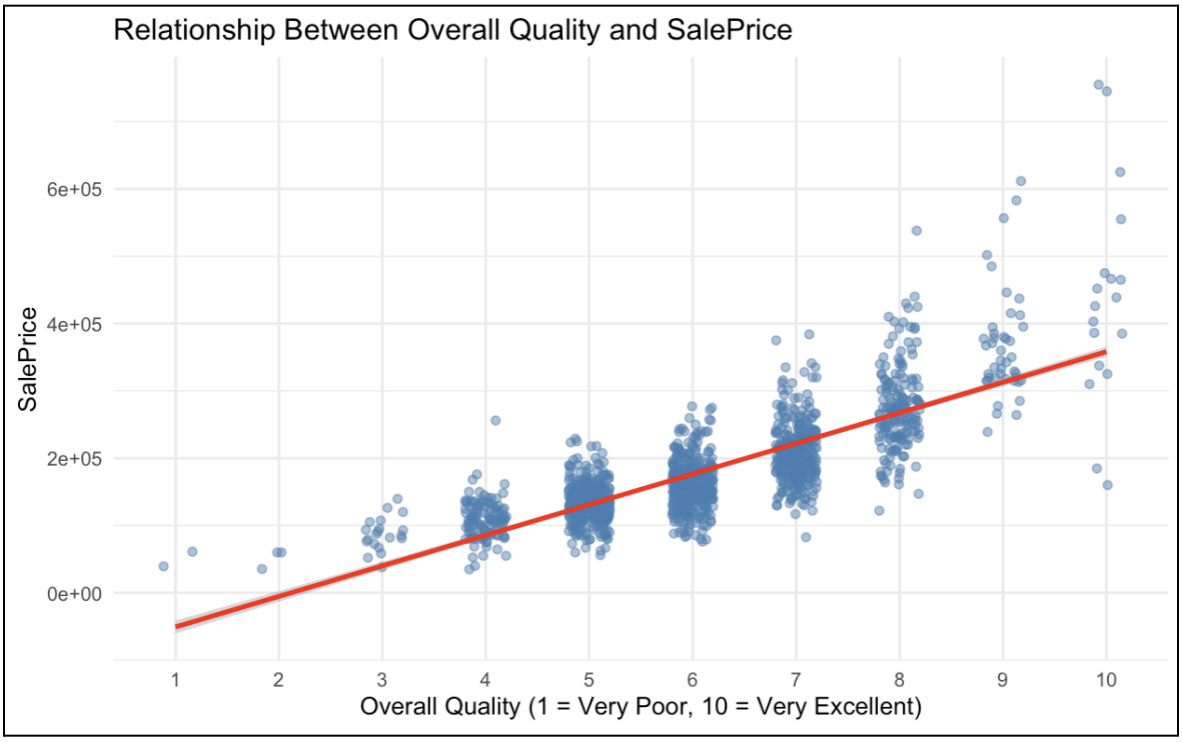
3.1.2. Plotting Numerical Variables

First, we have a look back at the correlation matrix between `SalePrice` and other predictors.

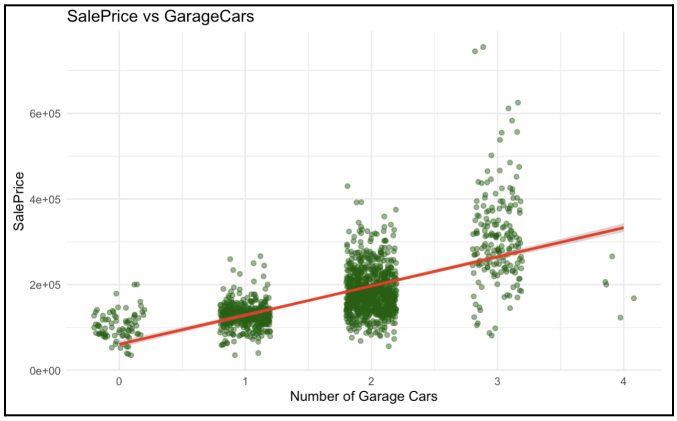
Top Variables Most Correlated with SalePrice

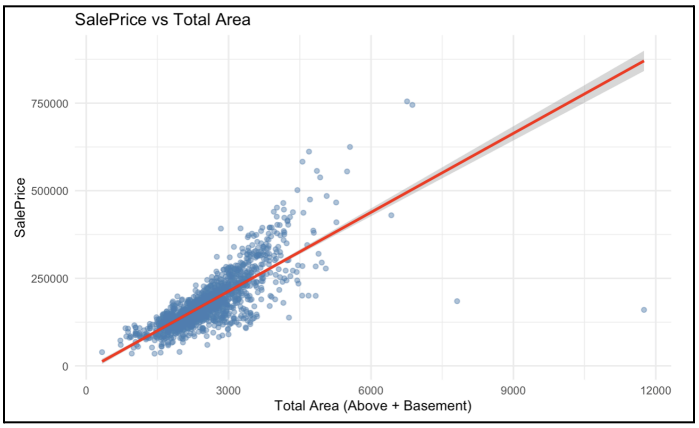
	Variable	Correlation_with_SalePrice
OverallQual	OverallQual	0.791
totalarea	totalarea	0.779
totalsf	totalsf	0.708
GarageCars	GarageCars	0.640
totalbaths	totalbaths	0.632
GarageArea	GarageArea	0.623
TotRmsAbvGrd	TotRmsAbvGrd	0.534
houseage	houseage	0.523
houseremodelage	houseremodelage	0.509
MasVnrArea	MasVnrArea	0.473

From the correlation matrix created above, we want to examine closer the relationship between `SalePrice` and its most correlated predictors (r > 0.65), including `OverallQual`, `Garagecars` and `totalarea`.



Clearly, there is a strong positive association between `SalePrice` and `OverallQual`, the higher the quality of home is evaluated, the higher the sale price of that home. It's also shown that there are less and less choice for the higher quality houses.

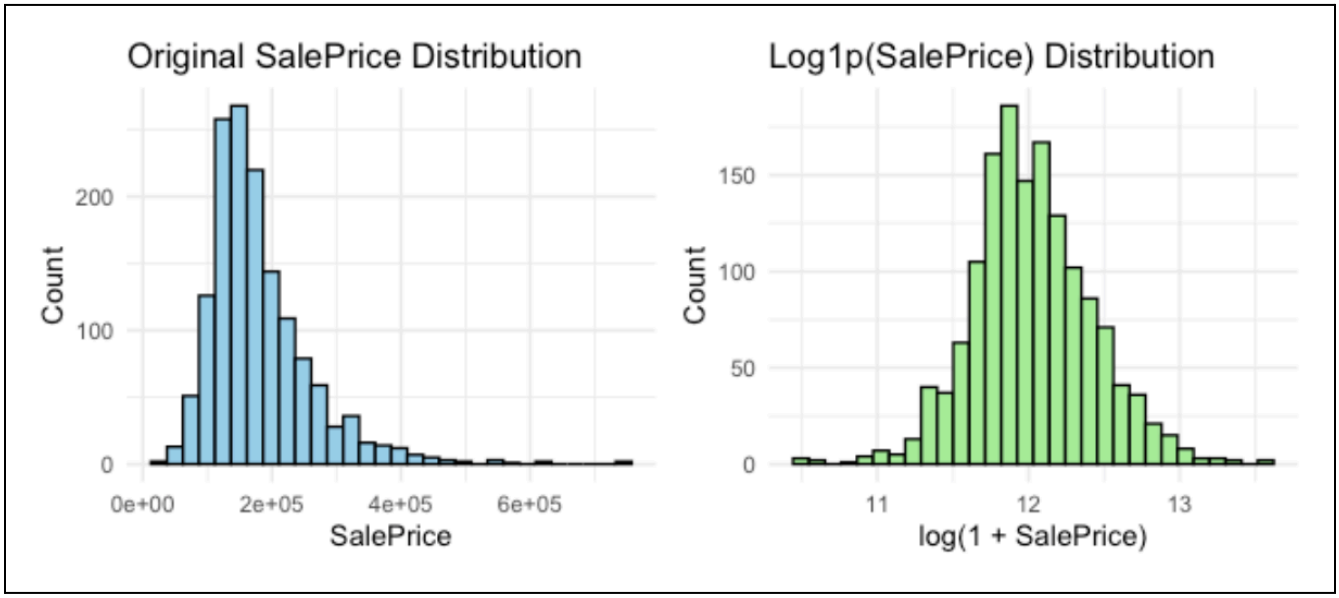




From the plots, `totalarea` - which measures the total square feet of living area (including both ground and basement) and `Garagecars` - the number of cars the garage can house are very strong indicator of `SalePrice` too. We see some very concerning outliers/high influencing points in `totalarea` plot.

3.1.3 Plotting Response Variable

Below is population distribution of our response variable - `SalePrice`. It is heavily right-skewed. So we want to see how the density might change if we apply a log transformation. Here I used the log1p (log(1+y)) instead of just natural log (log(y)) because it is adding 1 to safely transforming sale price that has zero values.



After taking log transformation, the distribution looks normal now. This can benefit many models, especially the parametric ones. Other benefits of using log transformation includes:
- Scale sensitivity: some models (linear regression, neural networks) are very sensitive to the scale of the target variable. Here log prices narrow the range.

- Homoscedasticity: Log transformation often makes the error variance more constant across the price range, which is an assumption especially for (generalized) linear models.
- Interpretability: In real estate, percentage errors are often more meaningful than absolute errors (a $10K error on a $100K house is more significant than on a $1M house).

In conclusion, I will use the transformed y for my model training.

3.2 Training/ Predictive Analysis

After data cleaning process, there are a total of 1460 observations and 194 predictors in our study. I utilize a 80/20 train-test split for training model as it's an adequate proportion (as compared to 70/30) to train models that need lots of data to learn well, all models will be trained on the same 1168 observations and tested on 294 observations. To evaluate the model's performance and assess its ability to generalize, a 5-fold cross-validation strategy was employed for all models except Random Forest, which used the "Out-of-Bag" Estimation method. All models will preprocess their feature inputs with centering and scaling because they are all on different scales and that will affect models that are vulnerable to comparing distance (SVM) and matrix multiplication (neurel nets). After scaling, the variables will have mean 0 and standard deviation 1.

3.2.1. LASSO Regression Model

- The shape of the training data is 1460*194, indicating pretty high feature-to-observation ratio. This can increase overfitting risk, especially if using complex models like neural networks or random forests. Also, computational efficiency will not be as good since training time increases with feature count and cross-validation will become much more expensive. So LASSO is a good algorithm to deal with this problem as its primary goal is to select features.
- Using LASSO, test RMSE = 0.1534. Cross-validation (CV) decided that the best alpha is 1, but best lambda = 0, meaning this model behaves like regular OLS (ordinary least squares regression), so no penalty was applied at all. I also tested with very small lambda (= 0.0001) to force some shrinkage always, but then the best lambda is still that smallest value. So original data already fits well with no penalty.

3.2.2. Elastic Net Model

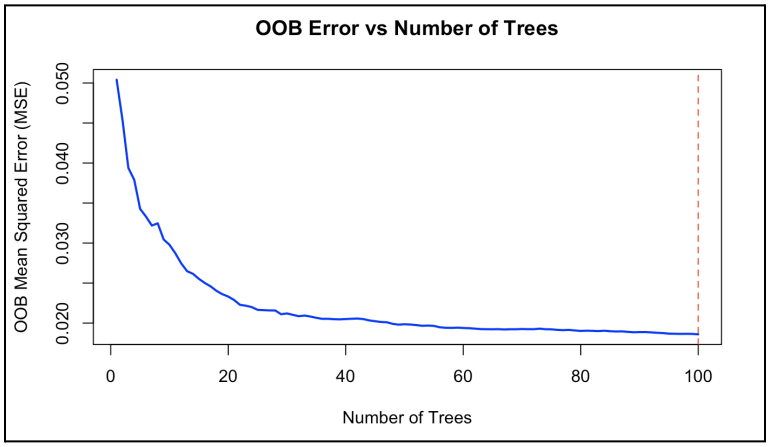
- I also want to try Elastic Net - the combination of the strengths of L1 regularization (which encourages sparse solutions by setting some coefficients to zero) and L2 regularization (which shrinks the coefficients towards zero). Tuning parameters were set the same as in LASSO.
- Using ElasticNet, test RMSE = 0.1434. Cross-validation (CV) results in best alpha is 0.1, lambda = 0.051.

3.2.3. Random Forest

- The third model that is used in this report is Random Forest. The reason to use RF is that there are many predictors with complex interactions that I'm not sure about and potentially non-linearity relationships between the predictors and the response in the dataset. I'm also concerned about overfitting of this data as there are so many predictors. RF allows ensemble of small decision trees that help reduce overfitting. Lastly, in EDA, we observed a lot of strong outliers which are concerning to the assumptions of building

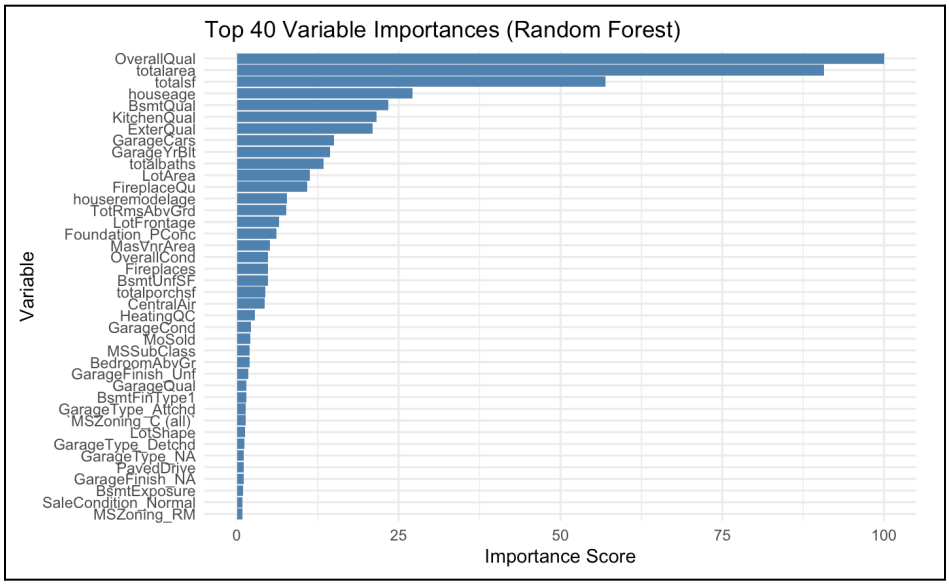
many models. And random forests usually can handle outliers very well. As stated, validation method that was used to estimate the model's performance is OOB - "Out-of-Bag" Estimation. In fact, we can skip splitting data into testing and training set here, but I would like to still separate them to compute test RMSE for model comparison later. Below is the function for building a random forest model.

- Using RandomForest, test RMSE = 0.1391. Cross-validation (CV) results in best mtry = 40, meaning best performance is reached when randomly selecting 40 features (columns) at each split of a tree. Later on, if we want to experiment tuning more mtry, it's beneficial to see when OOB RMSE stops improving.



From the plot, we can see:
- Early trees: error drops fast.
- Later trees: error flattens out (diminishing returns).
- Best spot: minimum point in curve, the elbow!

Also, after 100 trees, the OOB error stops improving much, so that gives insight about threshold for further tuning. Beside impressive performance, Random Forest can test which variables are the best in predicting house price, which are useful information to communicate and report to people not in Stats. Top 40 predictors are seen as below:



3.2.4. XGBoost

- Another potential model to fit is XGBoost (Extreme Gradient Boosting). It is a highly suitable model for predicting house prices because it is designed to work exceptionally well with structured tabular data. XGBoost also models non-linear relationships and complex feature interactions automatically. It also has built-in L1 and L2 regularization (via parameters like gamma, lambda, alpha), helping prevent overfitting on small datasets (but large proportion of predictors) like this housing data. With early stopping, XGBoost can automatically stop training when validation error stops improving, reducing training time and preventing unnecessary model complexity. In this experiment, I also utilize its built-in tools - early stopping to optimize training time and efficiency.
- I also fit an XGBoost with early stopping...
- Using XGBoost with no early stopping, test RMSE = 0.1188, best nrounds = 500. With early stopping, test RMSE = 0.1289, best validated nrounds (number of trees) is 133. I tried increasing the early_stopping_rounds and retuning other parameters in xgb_params but there is no improvement whatsoever. This implies that my model benefits from more trees, i.e growing up to 500 trees captured additional useful patterns and sometimes when early stopping stopped too early, results are not the best performance.

3.2.5. Support Vector Machines

- Next experiment model is Support Vector Machine with a radial basis function (RBF) kernel. According to my research, "rbf" usually performs the best in modelling non-linear relationships between features and the target variable so using the kernel is a safe choice. Another reason is SVMs are very effective in high-dimensional spaces (my data) and can handle both numeric and transformed categorical variables very well after proper preprocessing. Using the kernel trick to map the input data into higher-dimensional space, the SVM can separate complex relationships that traditional linear models (OLS) might miss - my first model. Lastly, by tuning parameters such as the cost (C) and kernel width (sigma), SVMs provide a good balance between model complexity and generalization, making them a strong candidate for structured regression tasks like house price prediction.
- Using Support Vector Machines, test RMSE is 0.2047. Best hyperparamater from tuning are C = 10, sigma = 0.01.

3.2.6. Neural Networks

- The last potential model is a single-hidden-layer feedforward neural network. It is a flexible algorithm that can approximate complex non-linear functions and learn the pattern in data very well thanks to its ability to back-propagate. In the context of house prices, where factors such as living ground area, overall quality, neighborhood, and various amenities interact in complex ways, neural networks can offer strong modeling capabilities because it can captures the noise very well. However, this is also a caveat, because when the dataset is a small-sized, Neural networks may risk overfitting.
- Using neural Network, test RMSE = 0.2216. It is found that best hyperparamters are: size = 3, decay = 0.1.

3.3 Optimization and performance evaluation

Before selecting the optimal model for prediction, selection criteria is considered. In a regression problem, main performance metrics is test RMSE (Root Mean Square Error). In the context of my data, test RMSE measures the square root of the average squared differences between the predicted house prices and the actual prices in my test dataset. So, lower values indicate better model performance.

As stated prior to training, the response variable (y) was transformed to log scale (ln(y)) in order to represent proportional errors rather than absolute dollar amounts, meaning a prediction error on a $100,000 house and a $1,000,000 house are weighted equally if they represent the same percentage difference. So the RMSE values in the table presented below (ranging from 0.1188 to 0.2216) can be transalted roughly to a 10%-20% error in the original price scale.

For example:

- For a $300,000 house, an RMSE of 0.1188 (XGBoost) means predictions typically fall within approximately ±$36,000 of the actual price
- For the same house, an RMSE of 0.2216 (Neural Network) means predictions typically fall within approximately ±$66,000

This variance in estimation is very significant in real estate contexts, where pricing decisions often rely on comparable sales that can vary by similar margins.

Comparison of Housing Price Prediction Models

Model	Training_Time_Minutes	Test_RMSE
LASSO	0.02 mins	0.1534
Elastic Net	0.16 mins	0.1434
Random Forest	0.20 mins	0.1391
XGBoost (W/o Early Stopping)	2.12 mins	0.1188
XGBoost (Early Stopping)	0.03 mins	0.1289
SVM	0.75 mins	0.2047
Neural Network	0.49 mins	0.2216

As seen from the summary table above, among the models evaluated, the XGBoost model (without Early Stopping) has the lowest test RMSE. However, its training time is significantly longer than the others. Thus, XGBoost (with Early Stopping) can be a better fit as it ranks second in test RMSE but is substantially faster than all the other models. Computation is resource is especially important if we have medium to big size sample, and that's usually what makes all the machine learning possible. So balancing between training speed and efficiency should be a priority. Therefore, XGBoost using Early Stopping would be my final chosen model. Its tuned hyperparamters is nrounds (number of trees) = 133.

Moving forward, if we want to interpret the predicted value/error from any model, we can just take the exponential of the predicted value/error.

IV. Conclusion

Overall, this project showcased the importance of thorough data cleaning, feature engineering, and feature/model selection when tackling real-world structured regression problems. Visualizations of numerical and categorical variables highlighted important trends, such as the positive relationship between overall quality, total area, and sale price, which then confirmed by Feature Importance graph from random forest.

In terms of modelling, multiple machine learning models were trained and evaluated, including LASSO, Elastic Net, Random Forest, XGBoost (with and without early stopping), Support Vector Machines, and Neural Networks. Each model was carefully tuned using cross-validation, and model performance was assessed through Test RMSE. Among these models, XGBoost with early stopping was chosen as the strongest predictive model, while simpler models like LASSO provided useful baselines with greater interpretability and still guarantees good performance.

These insights about the hyoperparameter tuning, data visualization and model evaluation can ultimately inform future efforts to refine predictions, optimize housing price estimations, and deploy predictive models in practice.