COMP4433 Data Mining & Data Warehousing

House Price Prediction

Competition Report

November 19th, 2024

Group: Team X

Group Member	Student ID
LEE Lee Ling	xxxxxxxxxx
HU Yuhang	xxxxxxxxxx
ZHOU Siyu	xxxxxxxxxx

Table of Contents

1. Introduction

The report basically includes the Kaggle Competition, House Price - Advanced Regression Techniques [1]. The objective is to introduce various methods for prediction, analyzing datasets, various feature engineering methods, different modeling methods, and analyzing improved methods. The analysis and methods mentioned in this report are implemented in the code and notebook link in *Appendix 2*. And the leaderboard score (performance of best prediction) is shown in *Appendix 1*.

2. Dataset Overview

2.1 Dataset Description

Training data consists of 1460 samples of data while the test data consists of 1459 samples. There are 81 attributes in total, including 36 numerical and 43 categorical attributes, as well as attribute ID and the target variable $SalePrice$. Attributes contain a variety of information about house property, which can be simply summarized into physical characteristics, facilities equipped, decoration styles and so on, contributing to the ultimate sale price.

2.2 Distribution of Saleprice

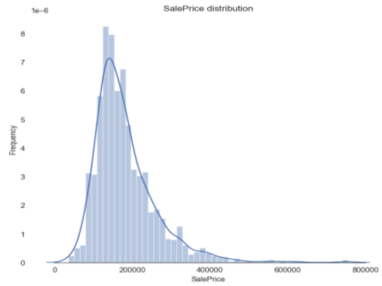


Figure 1 Distribution of SalePrice

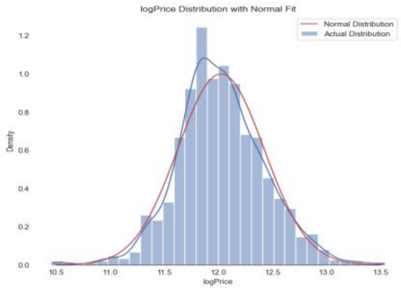


Figure 2 Log Transformation of SalePrice

As shown in Figure 1, the distribution of $SalePrice$ is distinctly right-skewed. Such a deviation indicates that though a majority of prices are below average, there are some residences with values much higher than normal, which can be noticed as outliers. Statistical modelling may be affected by this skewness so that adjustments are required to satisfy model assumptions. Meanwhile, the existence of outliers implies that those abnormally high-priced properties may have a negative impact on measuring central tendency. It may also present difficulties in modelling as most of algorithms are assumed to be applied to the normal distribution variables.

To reduce the consequences that skewness may bring, a logarithmic transformation to the target $SalePrice$ by using $\underline{\boldsymbol{np.log1p}}$ is adapted and its benefits are as follows.

- Applicable to models: Its application can help reduce skewness by distribution normalization, making the target variables more applicable to multiple algorithms in modelling that require normal distribution.

- Model Performance: The log transformation can reduce the distinct difference between normal data and outliners, which can improve the regression model's performance and produce more accurate predictions.

2.3 Missing Values in Dataset

Determining the extent of missing data is a crucial stage in exploratory data analysis (EDA) since it can have a big impact on how well predictive models perform. A number of factors can lead to missing data, including mistakes made during data collection and entry, respondent's neglect in services or chain effect due to the missing attributes (e.g., having no basement → missing value in $BsmtQual$, $BsmtCond$, $BsmtExposure$…).

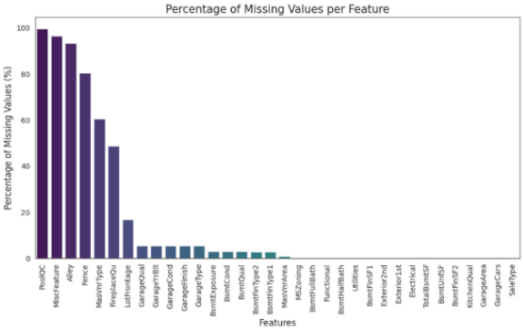


Figure 3 Percentage of Missing Values for each feature

Feature	Missing%	Feature	Missing%
PoolQC	99.66	MasVnrArea	0.79
MiscFeature	96.40	MSZoning	0.14
Alley	93.22	BsmtFullBath	0.07
Fence	80.44	Functional	0.07
MasVnrType	60.50	Utilities	0.07
FireplaceQu	48.65	Exterior2nd	0.03
LotFrontage	16.65	Exterior1st	0.03
GarageQual	5.45	Electrical	0.03
…	…	…	…

Table 1 Percentage of Missing Values of each feature

Features such as $PoolQC$, $MiscFeature$, $Alley$ and $Fence$ contain more than 80% of missing values. This data may disclose the fact that not all houses are equipped with corresponding facilities. In these cases, it is feasible to assume that missing values within these records indicate the feature does not exist. In contrast, Features with a low percentage (e.g. 0.03%) of missing values, such as $Electrical$, $KitchenQual$, and $GarageArea$, are probably absent because of mistakes in data collection and entry. Methods of handling missing values are varied and should be chosen with certain considerations, for example, data types and data formats. The most common methods include 1) filling in missing values with a specific value, 2) directly removing features with missing values, and 3) using models that can consider missing data. Further explanation of the procedure for handling missing values in this competition is given in *Part 3.2*.

2.4 Detect Outlier Through Distribution

In the analysis of outliers within the dataset, relationships between each numeric feature and the target variable ($SalePrice$) are investigated. Each feature is plotted against $SalePrice$, which allows for a clear examination of the data. This approach facilitated the detection of anomalies, as points that deviated significantly from the general trend can be indicated as potential outliers worthy of further investigation. Detailed steps for removing outliers are shown in *Part 3.1*.

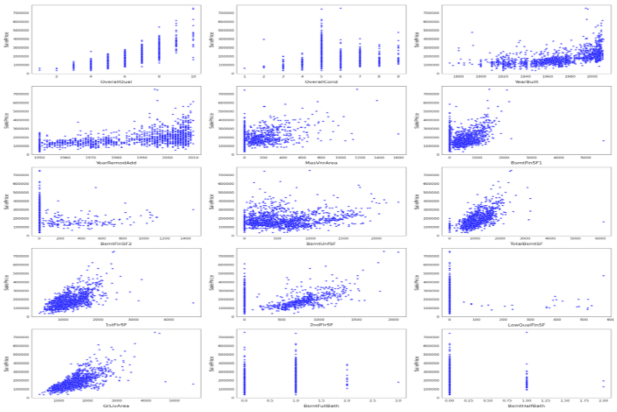
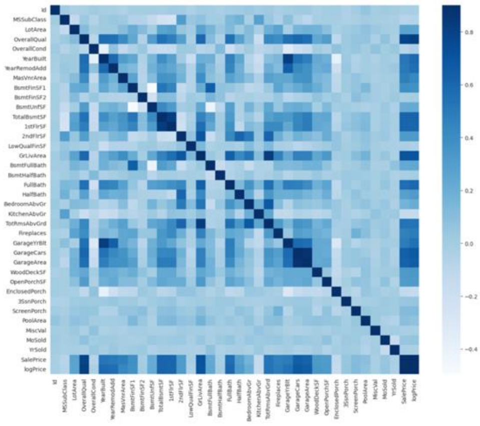


Figure 4 Detection of outliers in some features

2.5 Numerical & Categorical Feature Analysis on Sale Price



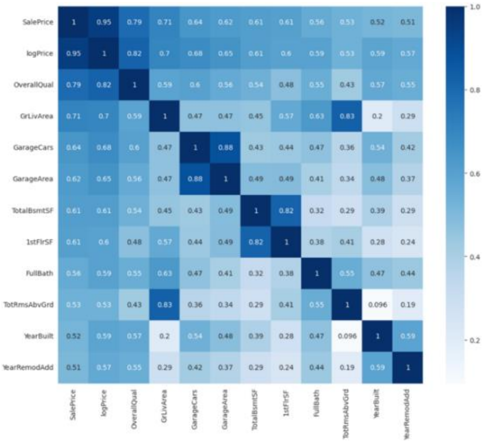


Figure 5 Numerical Feature Correlation

Figure 6 Top 10 Numerical Feature Correlation

When analyzing the dataset, several numerical features show a significant correlation with the *SalePrices*. Figures 5 and 6 provide a heat map of these features, which are crucial for forecasting house prices.

- ***Overall Quality (OverallQual):*** Figure 5 shows the strong positive correlation between overall house quality and its price. This suggests that houses with higher rates of quality sell for higher prices. It is reasonable as buyers are willing to pay more for a guaranteed house quality.
- ***Above Ground Living Area (GrLivArea):*** Figure 6 shows a positive correlation between the sale price and the living area above ground. It supports the notion that the bigger the house, the more expensive it is.
- ***Size of Garage (GarageCars & GarageArea):*** They are both the characteristics of a garage, referring to the number of cars it can contain and the garage's size respectively. Figure 5 illustrates the positive correlation between them and the sales prices. Many purchasers find a large garage to be a useful asset.
- ***Total Basement Area (TotalBsmtSF):*** Another important factor in determining the house price is the size of the basement. Figure 6 shows a positive correlation between

basement size and sale price.

- ***First Floor Square Footage (1stFlrSF):*** Figure 5 points out that a larger first floor is correlated with an increased sale price. Purchasers engaging in more home-held social activities or preferring larger spaces to play may be the cause of this.
- ***Functional Rooms (FullBath and TotRmsAbvGrd):*** Figure 6 illustrates how the number of bathrooms and rooms influences its sale price. They indicate convenience & privacy, and flexibility in room functions respectively.
- ***Year of Construction and Remodel (YearBuilt and YearRemodAdd):*** As shown in Figure 5, houses that are newly built or have recently renovated command higher prices since they offer modern facilities and require less maintenance.

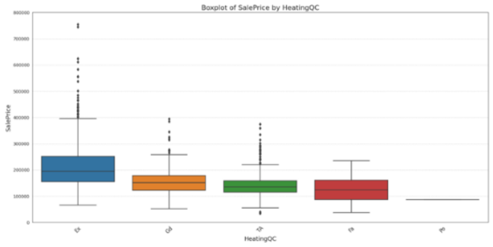


Figure 7 Heating Quality and Condition & SalePrice

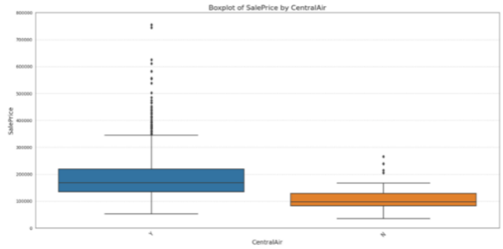


Figure 8 Central Air Conditioning & SalePrice

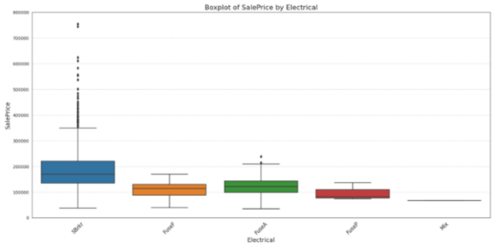


Figure 9 Electrical System & SalePrice

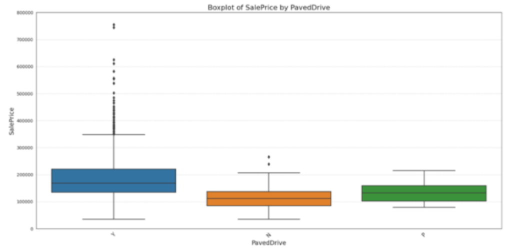


Figure 10 Paved Driveway & SalePrice

Besides numerical features, some categorical features also have a close relationship with *SalePrice*. 4 of them, *HeatingQC* *CentralAir* *Electrical* and *PavedDrive* are introduced. Because of their noticeable effect on the sale price and the existence of greater maximum sale prices in specific categories, these features are chosen. This raises the possibility that these particular features and increased house prices are positively correlated.

3. Feature Engineering Methods

3.1 Remove Outliers

From heat map, we know how features are correlated with sale prices. Here we take the two features *OverallQual* and *GrLivArea* that are most correlated with the selling price and remove their outliers. This can most effectively reduce the sensitivity of the model to outliers.

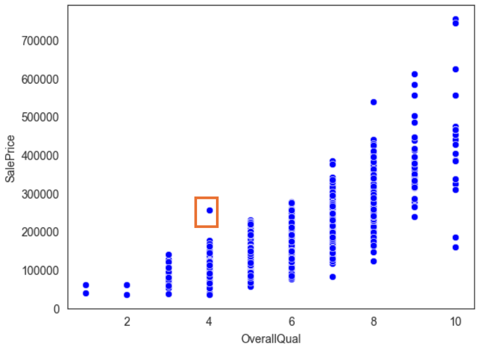


Figure 11 Overall Qual vs SalePrice

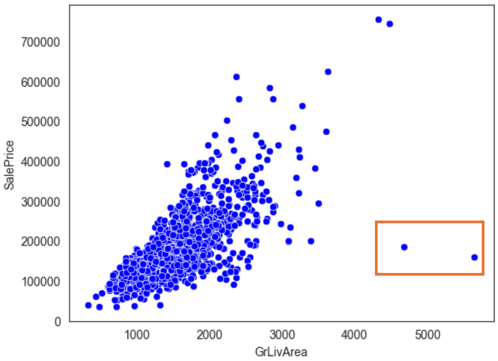


Figure 12 GrLivArea vs SalePrice

The above two figures show the distribution of sale price with different two features $OverallQual$ and $GrLivArea$.

- For Figure 11, we regard the point which $OverallQual$ is below 5 and $SalePrice$ is more than 200000 as an outlier.
- For Figure 12, we regard the point which $GrLivArea$ is larger than 4500 and $SalePrice$ is less than 300000 as an outlier.

3.2 Handle Missing Values

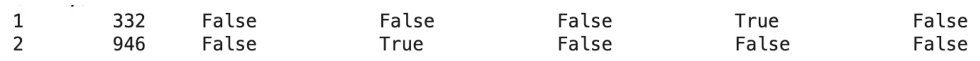
In *Part 2.3*, we find all missing values. In this part, we should handle them.

3.2.1 Fill in Missing Values

- Type 1: Special case: Assume typical unless deductions are warranted
  Features: $Functional$ (Fill in 'Typ')
- Type 2: Fill in mode
  Use mode to predict missing values for keeping the original distribution characteristics.
  Features: $Electrical$, $KitchenQual$,$Exterior1st$, $Exterior2nd$, $SaleType$, $Utilities$
- Type 3: Fill in 0 or 'None'
  - Fill in 'None' because NA in these features represent None.
    Features: $PoolQC$, $MiscFeature$, $Alley$, $Fence$, $MasVnrType$, $FireplaceQu$, $GarageType$, $GarageFinish$, $GarageQual$, $GarageCond$, $GarageYrBlt$, $BsmtQual$, $BsmtCond$, $BsmtExposure$, $BsmtFinType1$, $BsmtFinType2$
  - Fill in 0 because the features related to them are None or most of their values are 0.
    Features: $LotFrontage$, $GarageArea$, $GarageCars$, $TotalBsmtSF$, $BsmtUnfSF$, $BsmtFinSF2$, $BsmtFinSF1$, $BsmtFullBath$, $MasVnrArea$

3.2.2 Delete False Values (Train Data)

For some object features such as which are related to basement and garage, they should be all null values or all non-null values because if a house has no basement or garage, all features related to them will be $None$. Therefore, in this part we detect and delete rows which are not all null values or all non-null values in these features shown in Figure 13.



```
1    332    False        False        False        True         False
2    946    False        True         False        False        False
```

Figure 13 Rows should be deleted in training dataset

- The second column represents row number.
- The last 5 columns represent whether $BsmtQual$, $BsmtCond$, $BsmtExposure$, $BsmtFinType1$, $BsmtFinType2$ are null or non-null. True represents null. False represents non-null.

3.3 Manage Categorical Features

Use ***get_dummies*** method to manage categorical features helps convert categorical variables into a numerical format that machine learning models can understand. Its principle is one-hot encoding. Each category is converted into a separate binary column, ensuring that no information is lost. Unlike label encoding, it doesn't assume any ordinal relationship between categories. This can avoid ordinal assumptions.

3.4 Handle Skewed Values

The skewed features are characterized by an unbalanced distribution and cause the average to be influenced by larger or smaller values, making it deviate from the true center of the data. To improve the accuracy and stability of prediction, we should handle skewed values.

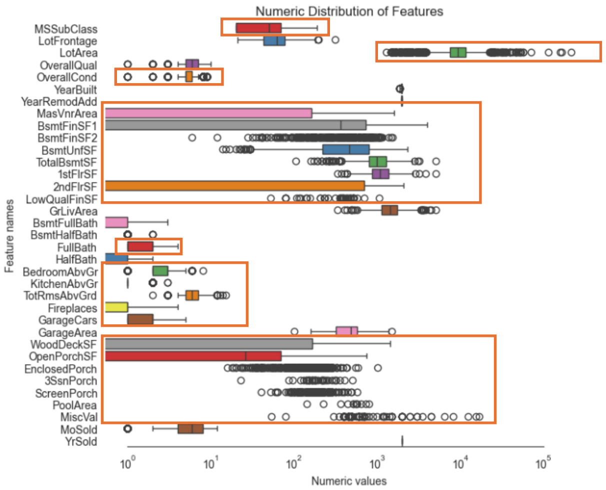


Figure 14 Numeric Distribution of Features

First, we find skewed numerical features with Skew > 0.5 here for threshold. There are 25 numerical features with Skew > 0.5. Then we use ***np.log1p*** function to normalize these skewed features. This function computes $log(1 + x)$ for each element x in the array. It is useful for handling skewed data because it reduces the impact of large values and can handle zeros without resulting in negative infinity, transforming the distribution to be more normal (Gaussian-like). Figure 15 shows the distribution of 25 features after normalization.

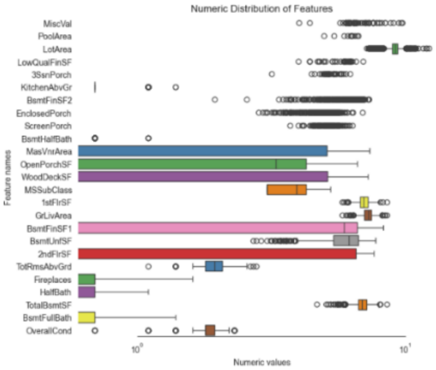


Figure 15 Numeric Distribution of Normalized Skewed Features

3.5 Create New Feature

Feature creation can unlock hidden information and improve predictive power. Here we create some new features according to the existing features.

- ***TotalSF***: the total sum of square footage including $TotalBsmtSF$, *1stFlrSF*,*2ndFlrSF*
- ***TotalSqrFootage***: the total sum of square footage excluding the basement
- ***TotalBathrooms***: the total number of bathrooms with half or full types
- ***HouseAge***: Age of the house when sold
- ***YearsSinceRemodel***: Age since last remodel
- ***HasPool***: whether the house has a pool
- ***Has2ndfloor***: whether the house has the second floor
- ***HasGarage***: whether the house has a garage
- ***HasBsmt***: whether the house has a basement

4. Modelling Methods

For model training part, our prediction methods were going to forecast house price with higher precision. In this part, we outline various methods implemented in the training of different models and analyze final predictions strategies.

4.1 Evaluation Metrics

The predicted sale prices are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price [1], which means, the primary evaluation metric in this project is Root Mean Squared Logarithmic Error (RMSLE). When $RMSLE$ is 0, it indicates predicted value perfectly fit observed values [2]. $RMSLE$ is calculated by following formula:

$$RMSLE = \sqrt{\sum_{i=1}^{n} \frac{(log(f_i+1)-\log(o_i+1))^2}{n}} \quad [2]$$

f_i is predicted value (future value or unknown result), and o_i is observed value (known result).

In the following part, cross-validation performance was evaluated using metrics such as Root Mean Squared Error (RMSE) to complement RMSLE and provide a comprehensive view of model performance.

4.2 Cross Validation

Cross-validation minimizes overfitting by simulating performance on unseen data. A **k-fold cross-validation** strategy was employed with $k = 5$. The dataset was split into 5 equal parts. In each fold, 4 parts were used for training, and 1 part was used for validation. The process was repeated 5 times, with each fold as the validation set, ensuring that model is evaluated robustly by testing its performance on multiple splits.

The RMSLE score with cross validation based on the training dataset was implemented. The score across different kinds of models was obtained after applying k-Fold cross-validation. The scores across folds for each model were visualized in following Figure 16.

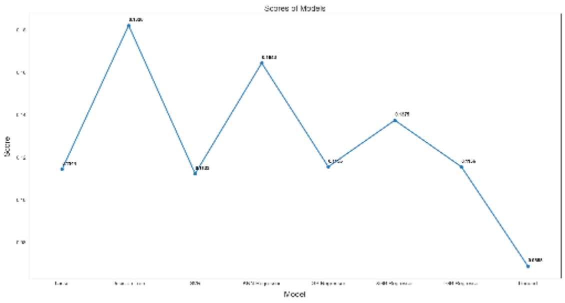


Figure 16 RMSLE Score of Different Models

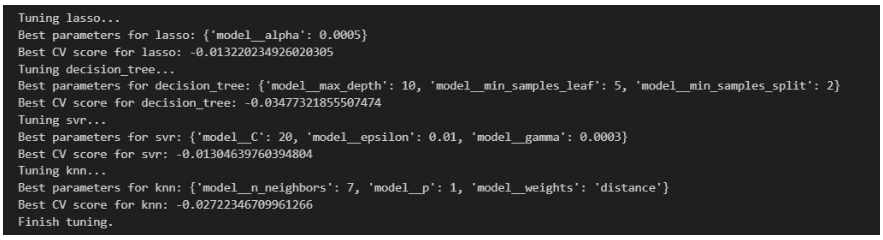
Models like Lasso, Support Vector Regressor (SVR), Gradient Boosting Regressor (GB Regressor), and LightGBM exhibited **low RMSLE** and consistent **high performance**. After **blending** model implements, the RMSLE score is **much lower**, as blended model only blends models with RMSLE score, showing the benefits of blending models. In some folds, XGBoost performed slightly higher RMSLE values, which is possibly due to underfitting or data-specific variability.

In conclusion, k-folds cross-validation evaluates the model on different data splits, ensuring robustness and standardizing evaluation across models.

4.3 Hyperparameter Tuning

Hyperparameter tuning optimizes performance by finding the best combination of parameters for each model. These parameters that cannot be learned during training and must be set before modelling. For this project, the $GridSearchCV$ method was employed to optimize the hyperparameters for the following models. For models with a limited number of hyperparameters, the $GridSearchCV$ method was used to exhaustively search over a predefined parameter grid.

A **5-fold cross-validation** strategy was employed to evaluate each combination of different parameters. The RMSLE metric was used as the optimization criterion. And in the following figures, the tuned parameters shown in Figure 17 are used for model training:



```
Tuning lasso...
Best parameters for lasso: {'model__alpha': 0.0005}
Best CV score for lasso: -0.013220234926020305
Tuning decision_tree...
Best parameters for decision_tree: {'model__max_depth': 10, 'model__min_samples_leaf': 5, 'model__min_samples_split': 2}
Best CV score for decision_tree: -0.03477321855507474
Tuning svr...
Best parameters for svr: {'model__C': 20, 'model__epsilon': 0.01, 'model__gamma': 0.0003}
Best CV score for svr: -0.01304639760394804
Tuning knn...
Best parameters for knn: {'model__n_neighbors': 7, 'model__p': 1, 'model__weights': 'distance'}
Best CV score for knn: -0.02722346709961266
Finish tuning.
```

Figure 17 Parameter Tuned for Different Models

4.4 Models

All models are trained through $make\_pipeline()$ function, makes it easy to apply the same preprocessing across all models. Before training models, the scaler $RobustScaler()$ was implemented as mentioned in Section 3.6. In this part, model used to predict house sale prices will be introduced as reference to other discussion posts [3][4], including its strengths:

Basic Models:

- $\boldsymbol{Lasso\ Regression}$: A linear regression model that adds L1 regularization to cost function, which is effective for datasets with many irrelevant features, reducing model complexity by eliminating irrelevant features.
- $\boldsymbol{Decision\ Tree}$: A tree-based regression model that splits data based on feature thresholds to minimize variance in target variable within each node. This model is useful for its interpretability but requires regularization to prevent overfitting.
- $\boldsymbol{Support\ Vector\ Regressor\ (SVR)}$: A regression model that aims to fit a function within a margin of tolerance around actual data. SVR could handle non-linear data effectively but is computationally intensive for large datasets.
- $\boldsymbol{K - Nearest\ Neighbors\ Regressor\ (KNN)}$: A non-parametric regression model that predicts value by averaging values of k nearest neighbors in the feature space. KNN is simple and intuitive but scales poorly with dataset size.
- $\boldsymbol{Gradient\ Boosting\ Regressor}$: A boosting algorithm that builds an additive model by sequentially decision trees on residuals of previous models. This model is powerful for capturing complex relationships but requires careful tuning.
- $\boldsymbol{XGBoost}$: An advanced gradient boosting framework optimized for speed and performance, handling non-linear relationships effectively and including regularization parameters to prevent overfitting.
- $\boldsymbol{LightGBM\ Regression}$: A gradient boosting framework designed for speed and accuracy, splitting tree leaf-wise rather than depth-wise, making it faster for large datasets, reducing memory usage and handling categorical features directly.

Ensemble Techniques

- $\boldsymbol{Stacked\ Model}$ **(ensemble)**: Combines predictions from multiple base models using a meta-model (SVR) for this project because SVR has lower score during cross-validation performance checking part. The $StackingCVRegressor$ function used in this project could leverage the strength of each model, and reduce individual model's weaknesses, improving generalization.
- $\boldsymbol{Blended\ Model}$ **(ensemble)**: A blended model aggregates predictions from multiple base models using weighted averaging (delete base model with relatively bad performance). The goal is to combine the complementary strengths of various models to improve accuracy and robustness, while reducing the weaknesses of individual models. Also, by changing the weight of different models, the performance improves as weight model with better performance (SVR and $Stacking$) is increased.

4.5 Model Performance

In our submissions, we evaluated various regression models to predict housing prices. The main objective of this competition is focusing on the impact of feature engineering based on model performance. The model performance with our log-transformation and feature creation was compared with the original score. The result explicitly shows that models with log-transformation and feature creation perform better than the original ones.

Model	***Scores***	***Scores After FE***	
Lasso	*0.12290*	*0.12755*	↓
DT	*0.19358*	*0.19325*	↓
SVR	*0.12284*	*0.12288*	↑*
KNN	*0.17308*	*0.19533*	↑
Gradient	*0.12625*	*0.12577*	↓
XGBoost	*0.16126*	*0.15694*	↓
LightGBM	*0.15654*	*0.16698*	↑
Stacking	*0.12285*	*0.12226*	↓
Blended	*0.12206*	***0.12142***	↓

Table 2 Comparison of Original Score and Score after Feature Engineering

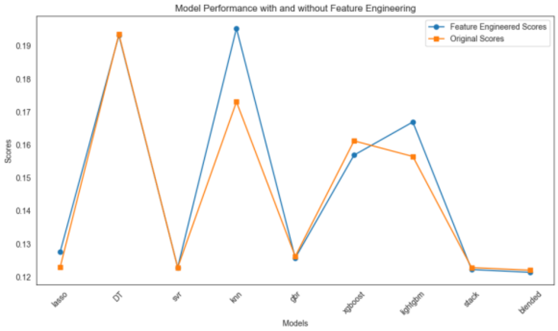


Figure 18 RMSLE Score with and without Feature Engineering

As above Table 2 and Figure 18 shows, feature engineering methods generally led to slight improvement in some models (e.g., $Lasso$, $Gradient\ Boosting$), but caused degradation in others (e.g., KNN, $LightGBM$). $Lasso$, $Gradient\ Boosting$ models are benefited from feature transformations that normalize or scale features. Feature engineering reduced $XGBoost$ score slightly but made it more consistent across runs.

For $\boldsymbol{SVR}$ model, the scores remain almost identical ($\mathbf{0.12288}$ vs. $\mathbf{0.12284}$), showing that support vector regression is relatively **robust** to data transformations. For

$\boldsymbol{Stacking}$ model, marginal improvement suggests that the model already effectively handles raw features. Models like DT ($Decision\ Tree$) show minimal changes. This is consistent with decision trees being inherently **insensitive** to feature scaling.

$\boldsymbol{Blended\ model}$ shows the best performance in both original (0.12206) and feature-engineered (0.12142) datasets. This indicates that blending is robust regardless of the data transformations. Also, by changing the weight of different models in this blended model, the performance improves.

5. Discussion Questions

5.1 Most Useful Feature Engineering Method

In our competition work, we use 5 methods to handle data in feature engineering. Among 5 methods, we think the most beneficial one is creating new features. Firstly, new features can capture information that original features might miss. For instance, the age of the house when sold may be more predictive than the year the house was built and the year the house was sold individually; and total bathrooms can capture more potential information on the quality of the house than the number of bathrooms per area respectively. Secondly, new features enhance model fit without significantly increasing complexity, unlike high-degree polynomials.

5.2 Works Done for Best Improvement

Among multiple submissions, the best improvement is the blended model with whole feature engineering process. The combined efforts in feature engineering, hyperparameter tuning ($GridSearchCV$), and ensemble technique ($Blended\ model$) led to a final leaderboard score of $\mathbf{0.12028}$, ranking the team as 303 in the competition.

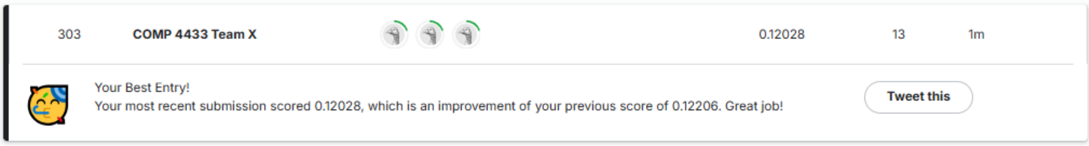
The feature engineering methods (skewed features handling, new feature creation) combined make predictions more valid as explained in *Part 5.1* above. Hyperparameter tuning ($GridSearchCV$ method) tunes the parameters by cross validating the model with different combinations of parameters as explained in *Part 4.3*. Also, the blended model weights different models, and improves model predictions and leverage strength of different models as mentioned in *Part 4.4* and *Part 4.5*.

Reference

[1] A. Montoya, “House Prices - Advanced Regression Techniques,” *kaggle.com*, 2016. https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview (accessed Nov. 17, 2024).
[2] “Root mean square deviation,” *Wikipedia*, May 03, 2024. https://en.wikipedia.org/wiki/Root_mean_square_deviation (accessed Nov. 17, 2024).
[3] lavanyashukla01, “How I made top 0.3% on a Kaggle competition,” *Kaggle.com*, Jun. 09, 2019. https://www.kaggle.com/code/lavanyashukla01/how-i-made-top-0-3-on-a-kaggle-competition (accessed Nov. 17, 2024).
[4] itslek, “Blend&Stack LR&GB = [0.10649] {House Prices} v57,” *Kaggle.com*, Mar. 21, 2019. https://www.kaggle.com/code/itslek/blend-stack-lr-gb-0-10649-house-prices-v57/script (accessed Nov. 17, 2024).

Appendix 1 Leaderboard Score

- Team Name in Kaggle: COMP 4433 Team X
- Ranking: 303
- Score: 0.12028



303 COMP 4433 Team X 0.12028 13 1m

Your Best Entry!
Your most recent submission scored 0.12028, which is an improvement of your previous score of 0.12206. Great job!

Appendix 2 Notebook Link

Notebook Link: https://www.kaggle.com/code/siyuuzoeyzhou/notebook-team-x