Predicting House Prices using Advanced Regression Techniques.

Kaggle playground competition: House Prices: Advanced Regression Techniques

The forecast of residential house prices helps real estate agents make decisions on prices, which in turn help their customers figure out the best selling prices of their properties. Even, the industry leaders apply the forecast of house prices differently. On the basis of such predictions, Zillow advises online sellers and buyers, and Airbnb guides hosts in the correct setting of prices for accommodations of all types. The integration of deep learning into the prediction of house prices marks a significant step forward in the application of AI in real estate, showcasing how historical advancements in AI can revolutionize contemporary challenges.

Deep Learning: A Quick Overview

Historical Context and Evolution

Deep learning is a revolutionary advance in artificial intelligence that has its roots in the most rudimentary first stabs at some of the computational and theoretical groundwork for efforts into neuroscience. Our journey starts with the perceptron, introduced by Frank Rosenblatt in 1958. Such a simple, primitive model was the first step toward simulating the functioning of the neuron.

But enthusiasm was quite short-lived: by 1969, the criticisms of Minsky and Papert pointed to some of its most limiting points, especially on the grounds of being unable to solve the problem of the exclusive-or circuit (XOR problem). Two of the most significant events in the 1980s, which were responsible for the revival of neural networks, include the creation of the backpropagation algorithm and the nearly parallel appearance of the Hopfield network, which proved that neural networks are good not only for pattern recognition but also for associative memory.

And during the 1990s, the addition of RNNs and, subsequently, long short-term memory (LSTM) networks would finally add temporal dynamic behavior and memory to neural network models, which had heretofore been agnostic of time. But it is in the early 21st century when really the explosion of big data and great progress in computational power (most especially GPUs) have made huge strides in shining a light on the efficacy of deep learning. Deep neural networks, meaning that their hidden layers had so many units, offered the promise to automatically find features and perform most of the difficult work involved in many classes of models.

It was an era of radical architectures like Convolutional Neural Networks (CNNs), which are good at processing visual data, and re-emergence of RNNs and LSTMs, which became pivotal in sequential data processing like language.

Fundamentals of Deep Learning

The deep learning constructs its architecture according to the organization of human brain structures and organizes the artificial neurons into a complex network. In a neural network, all respective neurons within the system receive input; these are then processed using a weighted sum and an activation function before passing this on to other subsequent neurons. The magic of deep learning is in the depth and complexity of such layers, enabling the network to learn hierarchical representations.

The core types of deep neural networks include CNNs and RNNs. One of the nicest properties of CNNs is that, through its unique structure of convolutional and pooling layers, it is capable of processing grid-like data (such as pictures) very well, so that spatial hierarchies are captured very effectively.

On the other hand, the latter is appropriate for sequential data (e.g., text or time series) in that they exhibit good performance for tasks with long-range dependencies; this is because they can carry information across time steps with the use of their internal memory.

Deep Learning vs Traditional Machine Learning

So, the deep learning spreads away from classical machine learning through several principal issues. Traditional algorithms—such as decision trees or Support Vector Machines (SVMs)—needed a feature engineering process, executed by human experts, and most of the time are limited to the complexity of high-dimensional data. Deep learning generalizes this to an unsupervised feature learning by extracting features at multiple levels, with higher-level features resulting from combinations of lower-level ones. This capability is one of the most crucial, for example, while dealing with unstructured data; relevant features might not be really clear at first sight, say, in images or text.

Second, deep learning methods are able to reach a data size and complexity at a scale where traditional methods often fail to do so. And this, however, is a double-edged sword, since the best models in deep learning usually require a colossal size of data.

Applications and Impact

The applications of deep learning are vast and transformational. In areas like the recognition of objects and the human face, the deep learning model has been at par or better than human performance. Models such as BERT and GPT, at the frontier of machine understanding and text generation, are enabling breakthroughs in translation, summarization, conversation, and much more. For predictive analytics—reaching from finance to health industries—to forecast trends and detect anomalies, today they need to find a happy medium with deep learning.

Challenges and Future Directions

It, therefore, has challenges on an equal measure with the many successes. These limitations come from the necessity for large and huge computational resources and large-scale datasets. The "black-box" nature raises key questions on the interpretability and trust of such models. This brings forward the ethical considerations, and particularly those related to biases and privacy, to bear in growing relevance as these models are deployed in reality. The future of deep learning is likely to focus on overcoming these challenges. The next wave of deep learning innovations will be, however, determined by the following: continued efforts pointing toward more efficient models that need less data and less computing power, research in 'explainable AI,' and both ethical guidelines and regulation. This evolving landscape holds promise for further advancements that might at last integrate concepts from cognitive science and neuroscience, further strengthening and robustening the AI systems.

Predicting House Prices with Advanced Regression

Project Overview

This project aims to predict the final price of a house from 79 explanatory variables summarizing almost any aspect of residential properties in Ames, Iowa. This is part of a Kaggle playground competition (https://www.kaggle.com/c/house-prices-advanced-regression-techniques). This project was analyzed in much detail using the data that was compiled by De Cock (2011). Kaggle is a platform that hosts competitions in data science.
Competitors are challenged to develop a model from the provided data and submit, after making predictions, the result to Kaggle. Typically, Kaggle provides a training dataset and a test dataset. Competents build a model from the training data and predict outcomes for the test data. Results from submissions are quantitatively assessed using particular metrics applied to the test data set. During competitions, a fraction of the test data set is used to compute a displayed score on the Kaggle Leaderboard; this is the public score. The final competition score—or, in this context, the private score—is computed from a complete test dataset. Throughout the model-building phase, most of the participants build up a private cross-validation score that matches with the competition metric and do it on their machine. This is often referred to as the local score. This project seeks to improve, most of all, the performance of the model based on the public Kaggle score.

Statement of the Problem

The main objectives of the project were building a model from the provided data and implementing an accurate prediction of the houses' prices—evaluation by Kaggle Leaderboard. Relevant problems to this are:

Therefore, one will always find it easy to explore what types of features are in the dataset, how much missing value exists, and if outliers exist, and look for further skewness in some features through exploratory data analysis.

Feature preprocessing: The preprocessing of the features, based on an overview of the exploratory analysis, may comprise feature transformation, data type conversion, outlier detection, and imputation for missing values if there are any.

Benchmark modeling: This type of modeling technique is used to set a up benchmark for future model enhancement using a standard method to create a model.

Model improvement: Improved model performance through parameter tuning and utilized ensemble learning techniques to create a collection of individual models for final submission.

Metrics

In this project, the evaluation metric to be used is Root Mean Squared Logarithmic Error (RMSLE). RMSLE is defined as:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(x_i+1) - \log(y_i+1))^2}$$

In this formula, Output represents the RMSLE score, n is the total number of observations, I+ are predicted house prices, C+ are actual house prices, and log($\mathbf{E}$) is the natural log of $\mathbf{E}$. The higher the RMSLE value, the greater the predicted and actual discrepancies in house prices.
Unlike Root Mean Squared Error (RMSE), which is predominant in most areas, RMSLE has less penalty even for a huge difference between the predicted and actual values. In RMSLE, the logarithmic approach is cautiously exercised to ensure that even in committing errors in predicting expensive houses and those, which are inexpensive, both will have an equally adverse effect and thus makes the procedure strong.

Analysis

Data Exploration

The training dataset is composed of both a mix of some categorical and numerical features and some missing values, having a total of 1460 observations of 79 features each, including the sale price. The test dataset has 1461 observations in each of its 78 features. We shall fill in the sale price for the test data using the trained model, which has been trained from the training data. Here is a histogram for the price of the house in Figure 1. The figure will show that the house price is skewed. Thus, the need for log transformation to normalize is a justification for the use of the RMSLE as our evaluation metric.

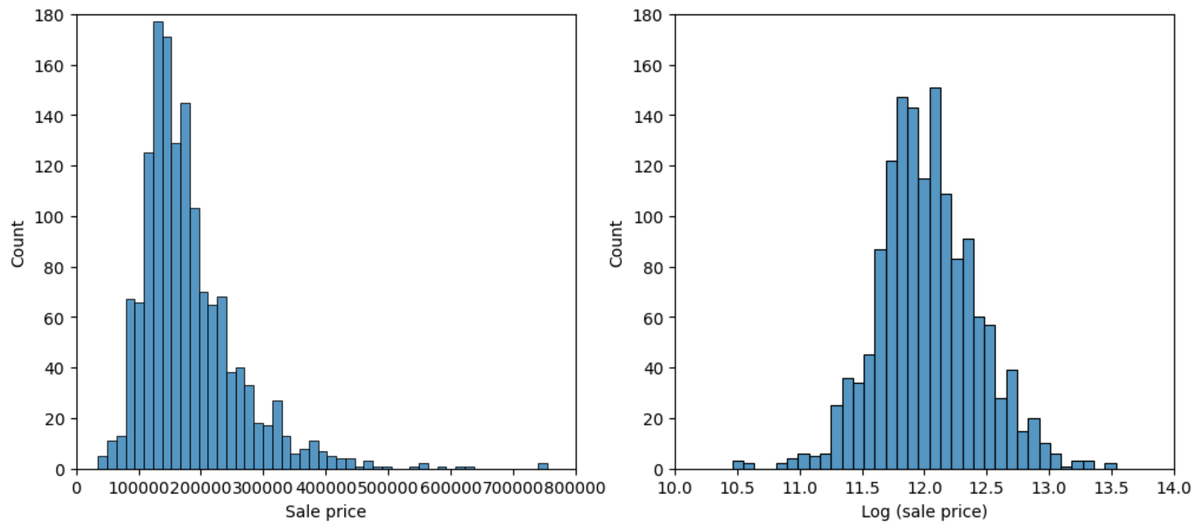
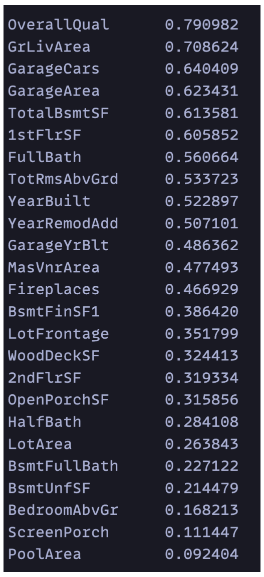


Figure 1: The histograms of the house sale price. (left) Original data; (right) Log transformed data.

Exploratory Visualization

Figure 2 shows the correlation matrix for numeric variables with the house sale. It shows that the number of variables has a strong relationship with the house price. In addition to this, I will explore the top 6 variables that have very high correlation power with the sale price in Figure 3.



```
OverallQual      0.790982
GrLivArea        0.708624
GarageCars       0.640409
GarageArea       0.623431
TotalBsmtSF      0.613581
1stFlrSF         0.605852
FullBath         0.560664
TotRmsAbvGrd     0.533723
YearBuilt        0.522897
YearRemodAdd     0.507101
GarageYrBlt      0.486362
MasVnrArea       0.477493
Fireplaces       0.466929
BsmtFinSF1       0.386420
LotFrontage      0.351799
WoodDeckSF       0.324413
2ndFlrSF         0.319334
OpenPorchSF      0.315856
HalfBath         0.284108
LotArea          0.263843
BsmtFullBath     0.227122
BsmtUnfSF        0.214479
BedroomAbvGr     0.168213
ScreenPorch      0.111447
PoolArea         0.092404
```

Figure 2: The correlation coefficients between some variables and the sale price

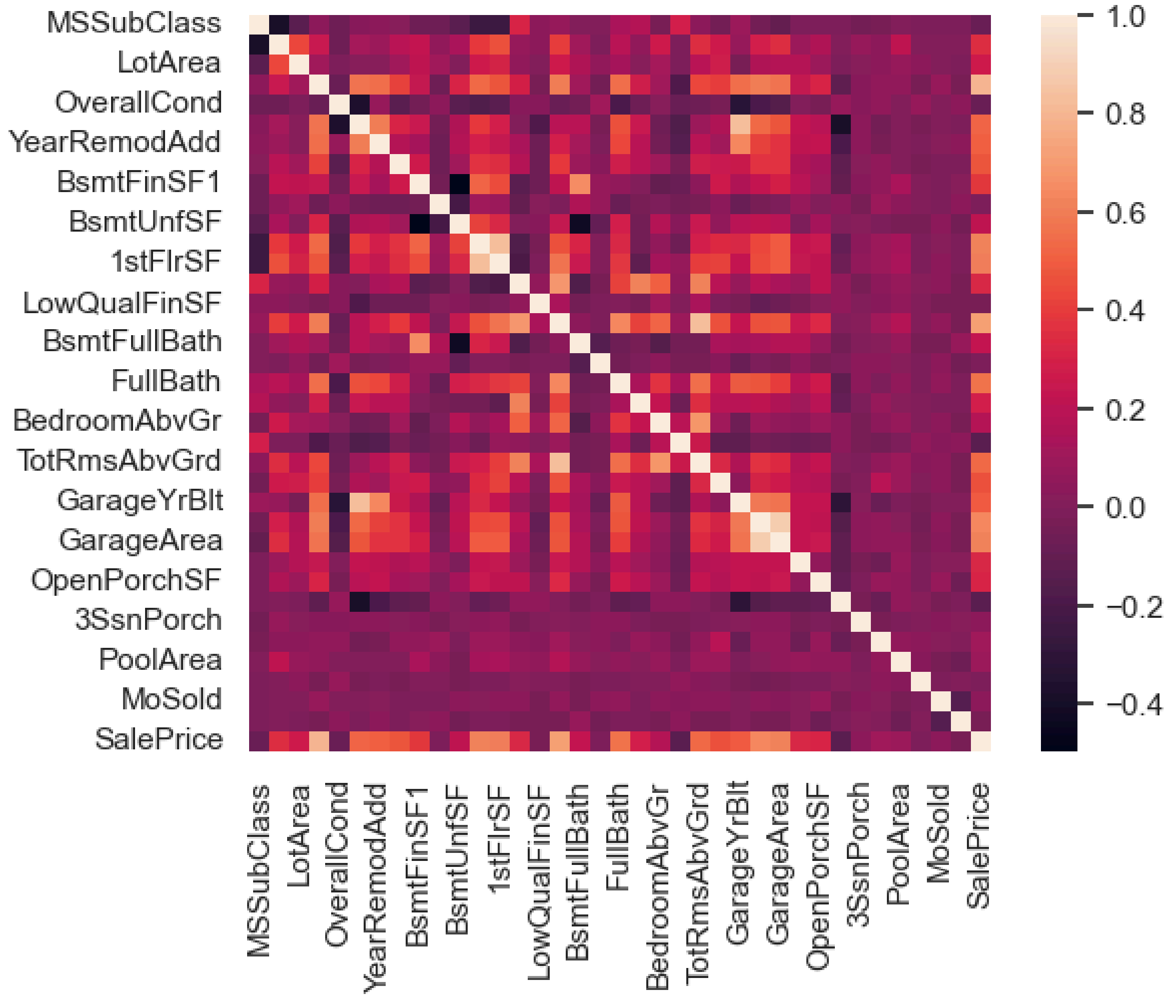


Figure 3: The correlation matrix between different numerical variables.

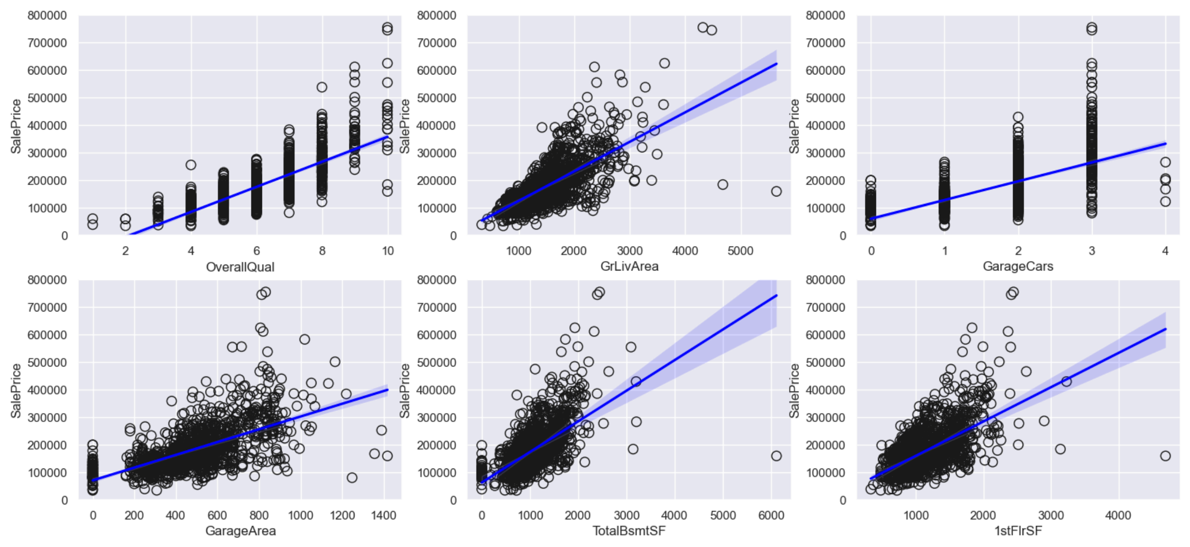


Figure 4: The scatter plot of the top 6 strongly correlated variables with the house price. The regression curve is shown in blue.

Example of the boxplot for the categorical variable between the neighborhood and sale price is as shown in the Figure 5. The pattern that is very evident and influences the price enormously is the feature of the neighborhood.

Where categorical features get into our machine-learning model, categorical variable takes some dummy variables that indicate value of 0 or 1, which indicates absence or presence of some categorical effect. Figure 5: Boxplot of sale price in different neighborhood.

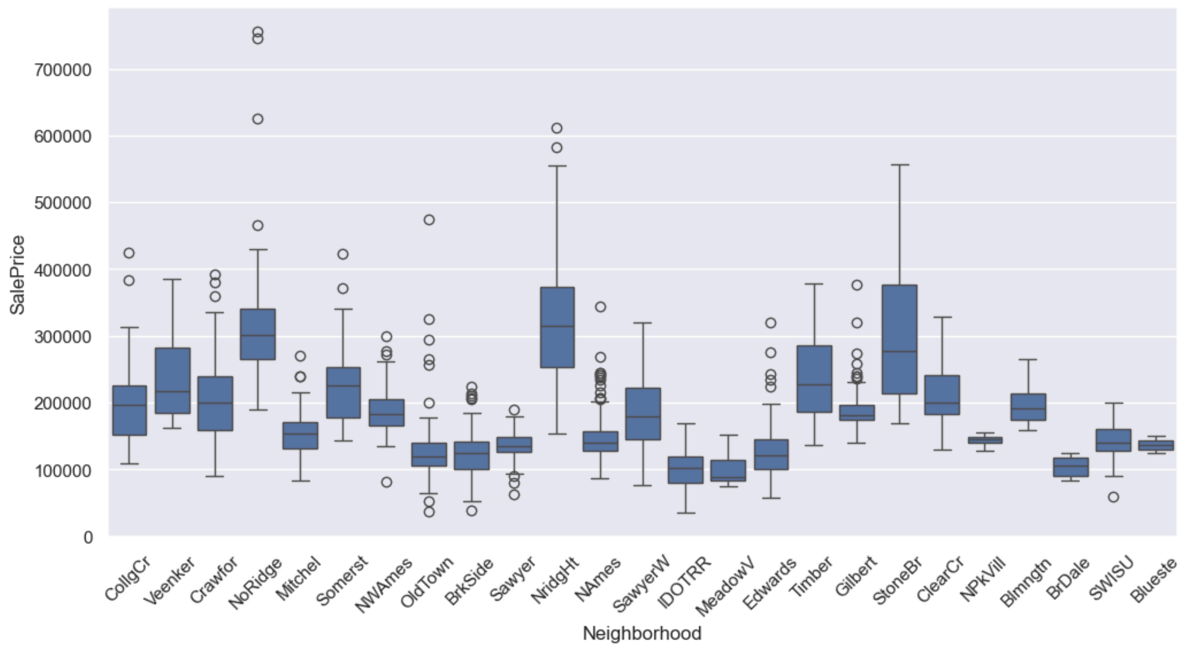


Figure 5: Boxplot of sale price in different neighborhood.

Algorithms and Techniques

So, one of the objectives in this exercise is to explore ensemble learning in order to see if a better score can be obtained among this advanced regression technique, among others. The techniques of ensemble learning develop a weighted combination of the base learners, sometimes referred to as a committee method (Murphy, 2012). The method applied during the project is known as stacking. Stacking is also familiar in its full name, "stacked generalization," and was first introduced by Wolpert in the year 1992. Since then, the use had expanded widely, being used by most of the participants in the competition of data science. One of the most famous cases is the winning solution of the Netflix Prize competition, BellKor's Pragmatic Chaos. At a very high level, the basic idea behind stacking is to use another model, or a "stacker," to blend all the previous model predictions in such a way that the generalization error is reduced. Here below is an example of a 2-level, 5-folds stacking approach in Figure 6:

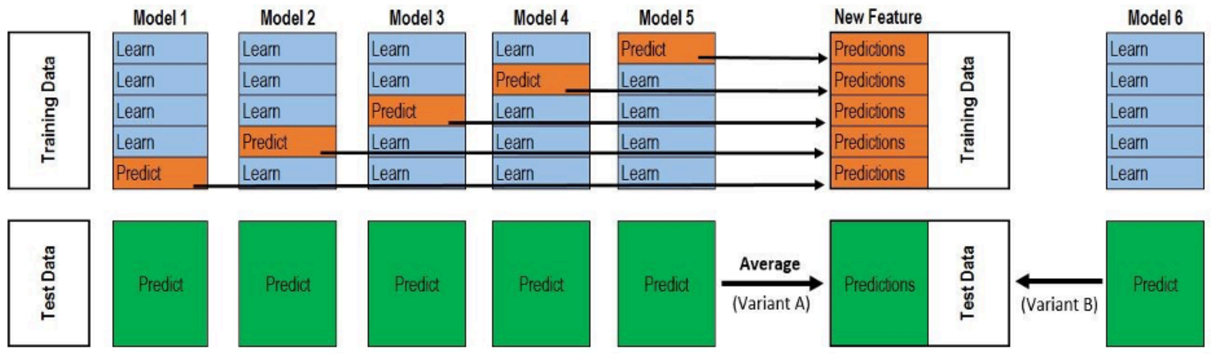


Figure 6: An illustration of stacking approach. (Credit: Kaggle forum)

First, divide the training data set with 5 folds. Iterate this 5-folds training data set.

Each iteration will fit a base model with 4 folds and predict on the holdout fold. Similarly, each base model also needs to provide a prediction for the whole test dataset in parallel. Having iterated over all the folds, we would have prediction for the entire training dataset for each model and have five copies of the prediction on the entire test dataset for each model. Lastly, the predictions in the training dataset are used as new features to train the second-level model, or stacker. It uses the average 5 test datasets' prediction results as the input for the trained model that will predict and output the final model.

Benchmark

Benchmark In this project, the benchmark model to be implemented is the Random Forest regression. Benchmark parameters are deemed to be the one from the scikit-learn package. It gave a local cross-validation score of 0.11997 and 0.14079 on the public leaderboard (rank 134 out of 4373 teams).

Methodology

A summarized illustration of the overall methodology and approach to modeling is derivable through the flowchart presented in Figure 7. The details pertaining to the approach shall be taken up under further subsections.

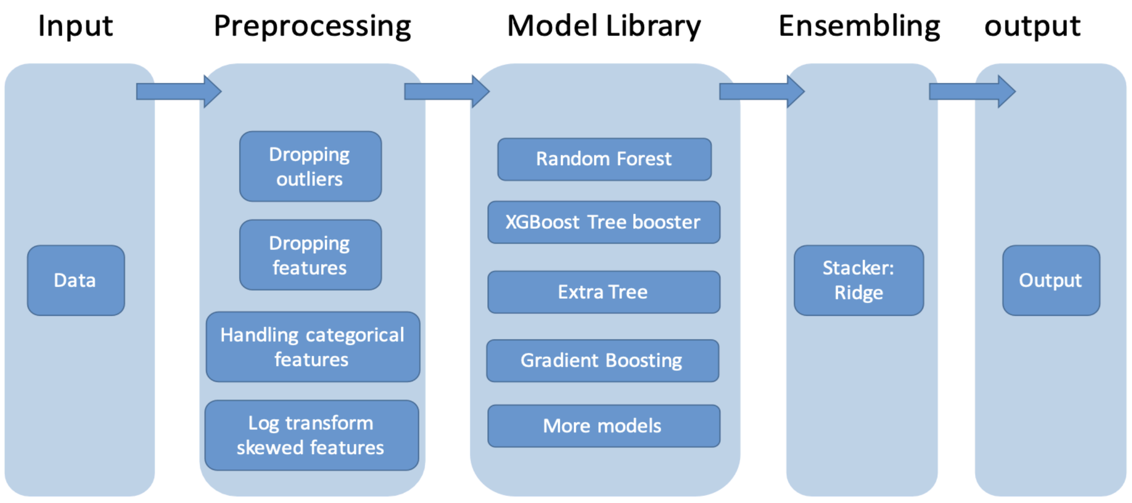


Figure 7: Flowchart of the modeling approach.

Data Preprocessing

1. **Outliers deletion:** For this question, all outliers any predictions by the benchmark varying from the actual house price by more than 12,000 - were coded. The 12,000 chosen here is quite arbitrary, but this approach actually increases both local score and public score by a bit in my explorations. All the outliers are dropped from the training dataset.
2. **Feature selection:** For selecting the features, I have simply proceeded by deleting the features with many missing values. In the categorical features, most of the missing values are for the features: Alley, FirePlaceQu, PoolQC, Fence, and MiscFeature. There are some highly correlated features with the sale price during Data Exploration. Some of the features lead to redundancy in variables and will be removed from the training data, while in this case, as it is evidenced by the past findings, it occurred with minimal loss in the local score/public score. Some other more advanced useful techniques of feature selection are, for example, the Boruto method (https://m2.icm.edu.pl/boruta/), though methods of feature selection are beyond the scope of this project.
3. **Log transform skewed features:** All features that have a skewness larger than 0.75 are log transformed.
4. **Handle categorical features:** The categorical features will be treated by the use of pandas "dummies" function, applied to the transformation of categorical variable conversion into dummy or indicator variables. Get the link: http://pandas.pydata.org below for your reference to the full Pandas information: (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html).

Implementation

First, I've trained and tuned 4 base models with hyperparameters before applying the ensembling learning approach, which includes the Random Forest Regressor, Extra Trees Regressor, Gradient Boosting Regressor, and Extreme Gradient Boosting Regressor. At the inception of the implementation, I use only 4 base models and vice versa into my final library model. The 4 models are: Random Forest Regressor, Extra Trees Regressor, Gradient Boosting Regressor, and Extreme Gradient Boosting Regressor. For my secondary model, I use Ridge Regression. The main intuition explains that the resulting predictions from the first-level model are used as new features to build a next-level model, where it makes its predictions accordingly. That's mainly why we are prone to including more models with ensembles, much as they may not perform that good—unless, of course, it is required to keep the base models unrelated.

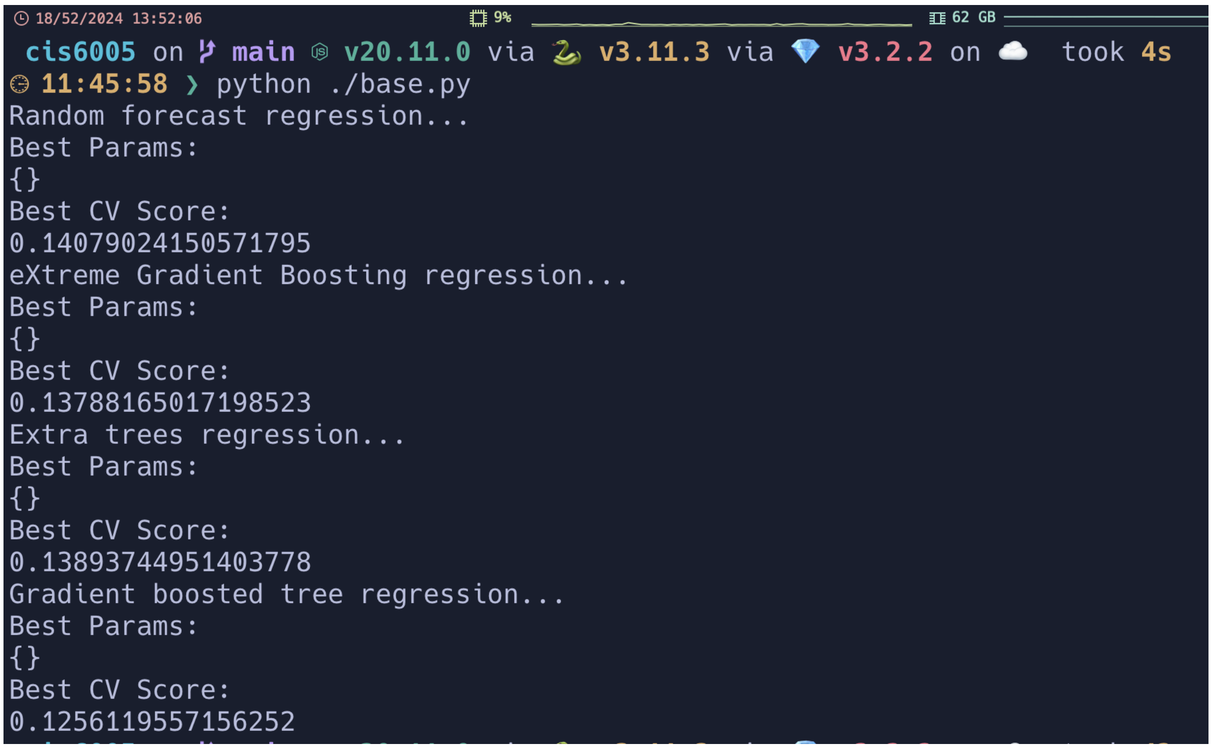
For my final, I used a way bigger library: Random Forest Regressor, Extra Trees Regressor, Gradient Boosting Regressor, Extreme Gradiant Boosting Regressor, LassoCV Regressor, K Neighbors Regressor, LassoLarsCV Regressor, Elastic Net Regressor, Support Vector Machine Regressor. Others are implemented from the scikit-learn package, which also includes the Extreme Gradient Boosting regressor from the XGBoost package. General ideas stick in the final model, but with more of a utilitarian general purpose base model library. A few will be run with exactly all the scikit-learn defaults and with parameters left at their default values.

There are two challenging parts during the coding process. That will take care of the feature in order to make a prediction for the final Second Level prediction. This is based on the prediction of the First Level base models. So, we also need to use the model prediction being a feature for the test dataset. Another similar thing would be the selection of base models. This is a really tricky, not fully answered question in this project because there are a lot of models to select. However, the rule of thumb from the investigation is that "choose the one that improves the local score" with a strong assumption that local and public score have good correlation.

Refinement

The refinement is conducted using the grid search technique.

The detailed grid range is shown in the submitted code file. Can see the local score before and after the grid search in figures 8 and 9. - For these models, I just used the default scikit-learn parameter. After taking out the final refined base models, we do the stacking ensembling to provide the final submission. Figure 8: The local cross validation score with default scikit-learn parameter values.



```
18/52/2024 13:52:06                    9%                         62 GB
 cis6005 on  main  v20.11.0 via 🐍 v3.11.3 via 💎 v3.2.2 on ☁️  took 4s
11:45:58 ❯ python ./base.py
Random forecast regression...
Best Params:
{}
Best CV Score:
0.14079024150571795
eXtreme Gradient Boosting regression...
Best Params:
{}
Best CV Score:
0.13788165017198523
Extra trees regression...
Best Params:
{}
Best CV Score:
0.13893744951403778
Gradient boosted tree regression...
Best Params:
{}
Best CV Score:
0.1256119557156252
```

Figure 8: The local cross validation score with default scikit-learn parameter values.

Results

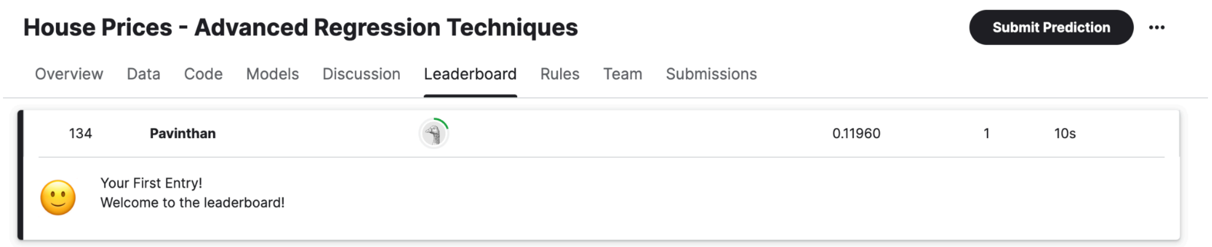
Model Evaluation and Validation

Another good observation for the ensembling model library process is that by aggregating more numbers of base models, it raises the potential of having good local validation scores and more Kaggle Leaderboard points. In model libraries, ensembles of different base models with their default scikit-learn parameter values are kept. With such a much larger library, I can now rank the top 134 among the over 4373 teams as of my writing this project report. The approach is supposed to be somewhat brute force, and the more correct way would be finding base models to be as uncorrelated as it gets. But I will cover these issues in the improvement section of model ensembling. Table 1 presents a local validation score and a Kaggle Leaderboard score for escalating modeling approaches

Model	**Local validation**
Random forecast	0.14079024150571795

Extra Tree	0.13893744951403778
Gradient Boosted Tree	0.1256119557156252
XGBbooster	0.13788165017198523
Stacking 4 best	0.11997511918438342
Final stacking	0.1241506530803208

Table 1: The local validation score for different model setup.



134	Pavinthan	0.11960	1	10s

Your First Entry!
Welcome to the leaderboard!

Figure 9: The Kaggle leaderboard at the time of my submission.

Justification

In summary, the Ensemble approach in this project improves the performance of the model from Benchmark Random Forecast Regression to the final model, named Ensemble Stacking approach. The final model ranked at 134/4373 in the Kaggle Leaderboard score and rank, compared to 1620/4373. The results are quite encouraging and the stage is open to a lot more thinking, learning, and exploration.

Conclusion

Free-Form Visualization

While developing the model, I will use the Kaggle Leaderboard score when building a model as accurate as possible in my final project visualization.

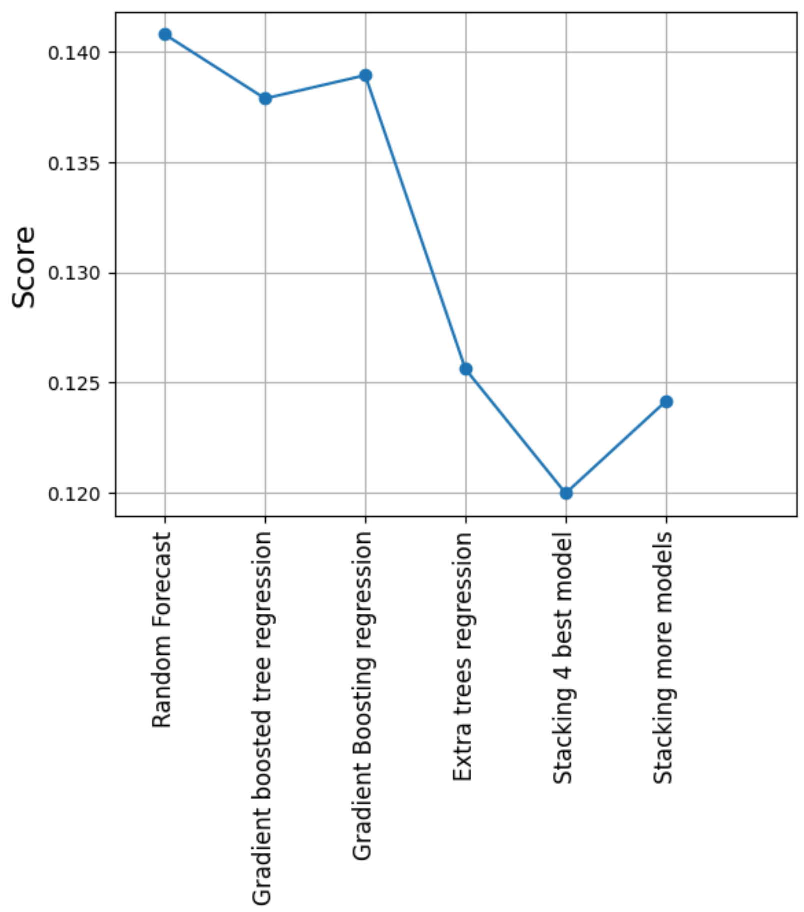


Figure 10: The Kaggle Leaderboard score at the different model development stage

Reflection

Starting off with explorative data analysis for this project: suppose there is some sketchy feature that is skewed, some features are categorical and ordinal, and some missing values need to be filled in.

Many of the data preprocessing steps were conducted to make the data ready for the machine learning modeling part. Further moving to the modeling part, I did train a number of base models and tuned their parameters.

Then applied ensemble learning on the stacking of all the base models to get the final submission. The hard part essentially relates to data preprocessing; e.g. it's hard to explain rigorously why I did such or such choice to enhancing the model. This calls for a lot of explorative testing and investigations to justify a choice based only on the local cross-validation scores. Another challenge emanates from the fact that hyperparameters for stacking models increase exponentially with the increase in the total number of models. The process is also very unwieldy, given that, for every k models, one must come up with a one-off pseudocode or piece of code chaining the output from all the models. Since the predictions by all the models can easily be chained in the wrong way, the guarantee of success is writing a pseudocode and checking it keenly before writing the production code.

Improvement

This project opens several avenues for future work. A big part is on preprocessing the data, which includes taking an advanced detection of an outlier, selection of features, and finetuning of the log transform threshold on the shifted variable, among others. Better creativity with feature engineering will also be very helpful. Another part is for the regression techniques. My gut feeling about this method of ensemble learning is that in case of a larger sizes of the pool of models, the score should improve. A winner of a Kaggle competition usually trains a few to a few dozen base models, up to (in some cases) many hundreds. For example: (https://www.kaggle.com/c/otto-group-product-classificationchallenge/forums/t/14335/1st-place-winner-solution-gilberto-titericz-stanislav-semenov). The trade-off here is about terms of deciding how much of your time and engineering efforts you want to spend compared to the tiny improvement being bell-curved. MLwave has a very nice suggestion at (http://mlwave.com/kaggle-ensembling-guide/) toward automating this whole work.

Reference:

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). https://search.r-project.org/CRAN/refmans/modeldata/html/ames.html
2. Murphy, K.P. (2012) Machine Learning: A Probabilistic Perspective. Cambridge, MA: MIT Press. https://probml.github.io/pml-book/book0.html
3. Friedman, J., Hastie, T., & Tibshirani, R. (2001) The Elements of Statistical Learning (Vol. 1). Berlin: Springer. https://pdfroom.com/books/the-elements-of-statistical-learning/Gk203vW3gpm