Evaluation of Feature Selection Methods on Housing Price Prediction

Daniel Son (50180108, d3m0b), Lynx Lu (34744136, h2r0b), Minghua Wu (13496138, s2h0b)
University of British Columbia
CPSC 532M

Abstract

We implemented three feature selection methods (BoLasso, ANOVA, xgboost) on top of a preprocessed dataset on housing price prediction. A custom implementation of the stacking ensemble method is used to evaluate the performances of the feature selection methods. And our work shows: 1) BoLasso method selected features that yielded the lowest validation error among the three methods 2) stacking ensemble method is more suitable as a benchmark for evaluation of the feature selection methods than the averaging ensemble method.

1 Introduction

Accurately assessing and predicting the value of living spaces is critical for many parties involved. Property investors require such information to form sound business strategies and accurately estimate the property value to streamline the sales and purchasing process. Easy access to such information would also allow residence and potential residence to correctly assess the value of their asset and make informed decisions. These decisions can include when to purchase or sell their homes, whether renovation or additions of certain features into the home would bring satisfactory return on investment and much more.

Papers on developing effective property sales price prediction models utilizing machine learning techniques are prevalent in academia, industry and data-science competitions. In data-science competition platform such as Kaggle, many teams have found great success and trained top models for housing price prediction through stacking various types of regression and classification models. However, there are very few top entries that have explored the possibility of including feature selection methods to improve the test error.

This paper explores whether including extensive feature selection techniques prior to prediction via stacking would improve the test error of the overall model. It will describe three different feature selection method and its potency in improving the test error. The dataset utilized in this paper is the Ames housing dataset compiled by Dr. Dean DeCock of Truman State University [1]. It will also adhere to the rules of the corresponding Kaggle competition "House Prices: Advanced Regression Techniques" [2].

2 Related Work

On Kaggle, an online data-science competition platform, ensemble methods like stacking is a very common practice in creating top regression and classification models. The Kaggle competition "House Prices: Advanced Regression Techniques", which utilizes the Ames housing dataset, is no exception. Majority of the top models tend to stack various base models like lasso, elastic net, gradient boosting regression (GBR) and XGBoost (XGB) [3, 4, 5]. However, these entries did not consider applying extensive feature selection methods to first determine the most relevant features to train with.

Few attempts at feature selection over the Ames housing database prior to stacking were attempted. These attempts include utilizing association approach or running cross validated models such as lasso or elastic net [6]. However, no entries have attempted to utilize more extensive feature selection approaches beyond this.

Extensive feature selection may help with improving the test error of the model because models trained over less features tend to overfit less (or have lower approximation error). However, it is worth noting that some of the common base methods utilized in the past have forms of regularization that can be selecting features while training, such as lasso and elastic net. But because this form of feature selection is not present in all the base model, some of the feature selection effect that few of these base models have may be lost during stacking.

3 Methods

The following section describes the dataset, how it was pre-processed, the methods utilized for feature selection and the predictor stacking approach.

3.1 Dataset

The Aimes housing dataset, published by Dr. Dean De Cock of Truman State University, is a comprehensive dataset on housing features and sales detail for residential properties in Aimes Iowa, US. The dataset includes information on homes which were sold between 2006 to 2010. In total there are 2930 examples, each with 23 nominal, 23 ordinal, 14 discrete and 20 continuous features. These features include properties of homes such as lot area, road access, neighbourhood and year built as well as sales transaction details such as year sold and sale price [1].

Due to moderate amount of examples available and many available features, machine learning methods have shown good amount of success in predicting housing prices in Aimes Iowa.

In accordance to the Kaggle competition rules, the first 1461 examples are utilized as the training dataset while the rest is designated as the testing dataset [2].

3.2 Preprocessing

Before feature selection, the dataset is pre-processed to fill in invalid or missing entries and to convert all the data in an acceptable numerical form. The whole process includes: null entry clean up, label encoding, skew correction and hot encoding.

3.2.1 Null Entry Clean Up

The dataset contains some missing values or null entries (entries as "NA"). For 'PoolQC', 'MiscFeature', 'Alley', 'Fence', 'FireplaceQu','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'MasVnrType' features, the NA entries imply that these features do not exist for the specified home. Therefore, it is simply replaced with "None". For missing numerical features such as 'GarageYrBlt' and 'MasVnrArea' (which are year garage was built and masonry veneer area), all null entries were replaced with 0. The null entries are, again, due to the feature not existing for the home in question.

3.2.2 Label Encoding

After null entry clean up, some of the categorical features were converted into ordinal features.

Features such as 'ExterQual', 'ExterCond', 'BsmtQual', 'BsmtCond', 'HeatingQC', 'KitchenQual', 'FireplaceQu', 'GarageQual', 'GarageCond', 'PoolQC' describe quality of the home feature in question. It ranges from none, poor, fair, average, good and excellent which are mapped to values from 0 to 5 respectively as integers. There are other features which also describe quality or degree of feature (ex. slope of property) of the home feature as categorical features but with different descriptors. These features were mapped in a similar way.

3.2.3 Skew Correction

Skew correction over ordinal and continuous features (including the sales price variable) were applied if its absolute skew score exceeded 0.75. The skew score is calculated by utilizing scipy.stats.skew function in python where 0 represents normally distributed dataset and highly positive or negative values represent highly right or left skewed distribution. The $log(1 + x)$ or numpy.log1p transform was applied to correct the skew of the distribution.

3.2.4 Hot Encoding

Lastly, the remaining categorical features (after label encoding) were transformed into binary features. This was achieved by creating new features for each classes of the categorical features then assigning 1 for the class occurring and 0 for the class not occurring.

Features such as 'MSSubClass', 'MSZoning', 'Alley', 'LotConfig', 'LandContour', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'Foundation', 'Heating', 'Electrical', 'GarageType', 'Utilities', 'Neighborhood', 'MiscFeature', 'SaleType', 'SaleCondition', 'Fence' underwent this transformation since its classes do not have any orderly relationship from its unique classes.

3.3 Feature Selection

3.3.1 BoLasso

BoLasso or bootstrap lasso is an ensemble feature selection technique. It trains multiple lasso regression models, each with its own standardized bootstrapped training dataset. For each lasso's training weights, features corresponding to non-zero weights are considered to be potentially significant. Across all the lasso's potentially significant feature sets, 90% intersect is computed and considered to be a relevant feature set. 90% intersect means if a feature in question is appeared potentially significant across 90% of the models, it is added into the final feature set. 90% intersect is utilized rather than the conventional intersect because relevant features may not be accepted if by chance at least one of the lassos did not consider it to be potentially significant. This is especially true if many lassos were generated, even though generating more base models to ensemble over should converge to a consistent and accurate output [7].

To search for the optimal regression coefficient λ, a dataset of only relevant feature set from each BoLasso is created then cross validated with an ordinary least squares model to compute the validation error. The λ value which yields the least error is selected as the optimal value, which was found to be 0.01. This was the result when iterating from $\lambda = 0$ to 0.1 with each BoLassos consisting of 20 lassos and running 20-fold cross validation.

Algorithm 1 BoLasso Cross Validating over λ

for $\lambda = 0$ to 0.1 step 0.005 **do**
  for $b = 1$ to 20 **do**
    Generate standardized bootstrap samples X^b and y^b*
    Compute lasso weight $\hat{w}^b$ from (X^b, y^b, λ)
    Compute support $J_b = \{j, \hat{w_j}^b \neq 0\}$
  end for
  Compute $J = 90\% \cap_{b=1}^{20} J_b$
  Compute 20-fold cross validated error E from OLS model†
  Save $E_{min} = E$, $\lambda_{min} = \lambda$, $J_{min} = J$ if $E < E_{min}$
end for
return λ_{min}, J_{min}
* The training dataset was standardized then bootstrapped
† The mean and standard deviation utilized to standardize the validation and training set for each iteration comes from training set only.

After finding the optimal λ, BoLasso without cross validating (at optimal λ) with 400 bootstrap models was performed to find the relevant feature set.

3.3.2 Xgboosted Regression (XGBRegressor)

Using Xgboost we can rank features based on feature importance score. Xgboost is a ensembled tree based model. While many top kaggle entries used this method for direct modelling of the data, there is no notable top examples of using XGBRegressor as a standalone feature selection method on top of model fitting. First XGBRegressor is fitted with training data, then the number of times a feature is used for splitting in the stumps is summed up and averaged for comparison across sum of all feature weights ("weight" is the number of times a feature is used across all ensembled trees in this definition). In order to have an idea of what the optimal number of top features to pick, a validation set is used to compare the RMSE to the number of top features selected. Using the elbow method, we then select the top 49 ranked features. This helps lower the amount of features that would be used to fit the model which can help reduce complexity and improve generalization.

3.3.3 ANOVA

ANOVA is a variance based feature selection method. Using F-ratio which is calculated by optimizing

$$f(w) = \frac{||Xw - \hat{y}||^2}{||y - \hat{y}||^2}$$

We model ANOVA of 'k' number of features in order to determine the optimal amount that would explain the most variance. A validation is set to first fit the feature selection model then fitting OLS with a separate test set to test the amount of RMSE for each hyperparameter value. It was found that approximately 50 features would be optimal using the elbow method.

3.4 Averaging and Stacking

Six base models are used in the averaging ensemble method. We implemented a simple averaging ensemble method which essentially takes the average of predictions from all base models as the final prediction.

Another ensemble method we implemented is stacking, which uses a meta-model to train on the predictions of the base models and also make final predictions based on the predictions from base models. In general, stacking of a number of under-performing or under-fitting base models would yield an overall better prediction accuracy [3]. The base models implemented are:

1. Linear regression with L2 loss and L1 regularization (Lasso)
2. Linear regression with L2 loss and both L1 and L2 regularization (ElasticNet)
3. Linear regression with L2 loss and L2 regularization (Ridge)
4. KNN Regression
5. lightGBM
6. Random Forest Regression

And the meta-model is linear regression with L2 loss and L1 regularization (Lasso) model.

4 Experiment and Analysis

4.1 Base models, Averaging and Stacking

For the base models, sklearn's implementation of Lasso, ElasticNet, Ridge, KNN Regressor, and Random Forest Regressor and Microsoft's implementaion of lightGBM were used as our base model implementations. The stacking model is our custom implementation, and thus to verify the performance of our stacking implementation, the pre-processed training dataset is used. Table 1 shows the root mean squared error (RMSE) of the base models. The RMSE of our stacking model implementation with the preprocessed training dataset can be found in Table 2, and the RMSE validation errors of our simple averaging ensemble model can be found in Table 3.

With 4-fold cross validation, our stacking model achieved an average validation error of 0.10562, which is lower than the validation error of any base model, and this shows that our stacking model

is effective and indeed outperform any of the base models. However, because some base models perform noticeably worse than other base models, the averaging ensemble model is also affected and yielded a validation error that's even worse than many of the base models [3]. And thus, the averaging ensemble model should not be used as the benchmark to evaluate the effectiveness of feature selection methods.

Table 1: Base model performace with preprocessed training set

	Lasso	ElasticNet	Ridge	KNN Regression	lightGBM	Random Forest Regression
training error*	0.10298	0.09955	0.09605	0.25888	0.06969	0.05621
validation error*	0.12601	0.12492	0.13144	0.28714	0.13393	0.14985

* All errors are calculated as root mean squared error (RMSE)

4.2 Evaluation of feature selection methods

Using the three feature selection methods (BoLasso, ANOVA, xgboost), the preprocessed training set is reduced by removing the features that are deemed not significant in the feature selection processes, and three new training sets with only selected features are generated. The BoLasso feature selection method selected 37 features out of the 243 from the preprocessed set while the ANOVA method selected 50 features and xgboost method selected 49 features.

The stacking model is used to evaluate the new training set produced from the feature selection methods, and the results are shown in Table 2. The evaluation used cross validation with 4 folds and the evaluation was conducted multiple times which showed consistent performance. As shown in Table 2, using the same stacking model with 4-fold cross validation, the preprocessed training set which has 243 features produced a RMSE validation error of 0.10562, which is the lowest among the four dataset. The dataset with 37 features selected by BoLasso method produced the next lowest validation error, and the next lowest is the dataset selected by xgboost method which selected 49 features. The dataset selected by the ANOVA method, which has 50 features, yielded the highest RMSE validation error at 0.12260. This may be attributed to the large number of binary type input data which may influence the variance calculations more than other feature selection methods.

Table 2: Evaluation of feature selection using a Stacking model

Stacking	Training dataset			
	preprocessed	BoLasso	ANOVA	xgboost
# of features	243	37	50	49
training error*	0.09422	0.10464	0.11010	0.10366
validation error*	0.10562	0.11138	0.12260	0.11471

* All errors are calculated as root mean squared error (RMSE)

Although the three datasets with selected features all had higher RMSE validation error, the errors are quite close to the one produced with all 243 features. With only 37 selected features, the BoLasso method significantly reduced the number of features while keeping the validation error very close to when all the features are kept. This indicates that other than many features that are not useful, there are some features that positively contribute to prediction accuracy.

Table 3: Evaluation of feature selection using a Averaging model

Averaging	Training dataset			
	preprocessed	BoLasso	ANOVA	xgboost
# of features	243	37	50	49
training error*	0.14732	0.14783	0.15165	0.15241
validation error*	0.17440	0.16965	0.17864	0.17977

* All errors are calculated as root mean squared error (RMSE)

However, it should be noted that this validation error ranking of the four data sets is different when we changed the ensemble method from stacking to averaging. Table 3 shows the training and

averaging results using our implementation of a simple averaging ensemble method with 4-fold cross validation. The features selected by BoLasso produced the best accuracy (lowest validation error) while xgboost produced the lowest performing features. Nevertheless, as we mentioned that because the averaging ensemble model is easily affected by under-performing base models, it is not suitable as the benchmark for testing the effectiveness of feature selection methods. Thus, we would not evaluate or recommend feature selection methods based on the errors from the averaging ensemble model.

5 Conclusion

As can be seen, the feature selection methods presented in this work were not successful in lowering the amount of error with all base-models. However, given the amount of reduction that was made possible by the top model, BoLasso, while having fairly similar results to the preprocessed data, there is potential benefits in terms of computation overhead reduction to be seen. Given a much larger housing data set with more features, feature selection with a stacking model may be good in reducing the dimensionality of the problem to speed up computation. However, upon using an averaging model, we see an improvement in BoLasso feature selection from the original preprocessed data.

In addition, we found that the stacking ensemble method with Lasso as the meta-model not only outperforms the base models, it is much more robust against underperforming base models compared to the averaging method. Combining the feature selection method and stacking of simple base models, there is more potential benefits in terms of reduced memory usage and reduced computation overhead. Further exploration of using such model would be recommended to see if further tuning can be made to improve upon the preprocessed dataset.

Acknowledgments

We would like to express our gratitude to Mark, Mike and all the teaching assistants for your wonderful teaching and helps throughout the course.

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] Kaggle, 'House Prices: Advanced Regression Techniques', *Kaggle* 2011. [Online] Available: https://www.kaggle.com/c/house-prices-advanced-regression-techniques. [Accessed: 05- Nov- 2018]

[3] Serigne, 'Stacked Regressions to predict House Prices', *Kaggle* 2017. [Online] Available: https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard. [Accessed: 05- Nov- 2018]

[4] Patel, A.G, 'Stacking House Prices - Walkthrough to Top 5%', *Kaggle* 2018. [Online] Available: https://www.kaggle.com/agodwinp/stacking-house-prices-walkthrough-to-top-5. [Accessed: 05- Nov- 2018]

[5] Gehsbargs, A., 'Top 10 (0.10943): stacking, MICE and brutal force', *Kaggle* 2018. [Online] Available: https://www.kaggle.com/agehsbarg/top-10-0-10943-stacking-mice-and-brutal-force. [Accessed: 05- Nov- 2018]

[6] Laurenstc., 'Top 2% of LeaderBoard - Advanced FE ', *Kaggle* 2018. [Online] Available: https://www.kaggle.com/laurenstc/top-2-of-leaderboard-advanced-fe. [Accessed: 05- Nov- 2018]

[7] Bach, F.R. (2008) Bolasso: Model Consistent Lasso Estimation through the Bootstrap. *ICML '08 Proceedings of the 25th international conference on Machine learning*:33-40