## Predicting House Prices with Machine Learning

**Team:** Roberts Kustavus (519, Yang’s cohort),Yueqi Wu (419, Pooja’s cohort), Zhanqi Wang (519, Yang’s cohort)

**Project Mentor TA:** Pooja

### Abstract

To develop a model that is able to predict house prices based on house features and to facilitate easier purchasing and research in the real estate market, we first ran linear models - Lasso, Ridge, Linear regression on the original and PCA/log transformed dataset. Then, we ran non linear models and constructed an ensemble stacking these models to improve generalization performance. Our final model achieved a test RSME of $24,644. We also ran classification models on the dataset with labeled house price level.

**Github Link to Code:** https://github.com/RobertsKustavus/CIS419

### Introduction

Real estate prices affect people’s daily lives. Buyers are concerned about whether the price of a house could be justified by its attributes and whether the sale price reflects the dynamics in the housing market. Policy makers are also concerned about how their urban policies would affect the house prices and whether those real estates would still be affordable. The problem being a part of an established Kaggle challenge will allow us to learn and gain practice in machine learning while being able to gauge our results in the context of the other participants.

### Related Prior Work

Kaggle challenge*:* our exploration is grounded in the Kaggle House Prices - Advanced Regression Techniques challenge. It is based on the Ames Housing dataset[1] which we will use as the main data for training and testing our models. Currently 6780 teams have joined the competition.[2] We did a survey of notebooks in the code section of the challenge and the dataset [3]. Many of the notebooks that did not use an ensemble of different models used normalized linear regression and random forests regression and used hand selected features[4-11]. Notebooks containing ensembles usually consisted of L1, L2 regularized linear regression, Gradient Boosting regression, XGBoost and some other model depending on the notebook.[12-18]. From the notebooks we reviewed only one used SVR as a part of their ensemble and in that one the ensemble weights were applied manually [12], so we want to explore the use of support vector machines further.

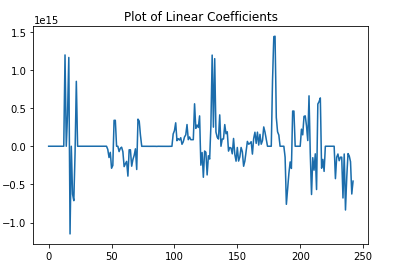
Ames Housing Dataset and stacking: Outside of the posts on Kaggle we also reviewed some academic work on the topic. There is a vast body of work in the field of machine learning that is based on the Ames Housing dataset. We focused on articles that relate to ensemble learning and advanced regression. Shahhosseini, Hu and Pham [19] tune parameters for an ensemble of L1 regression, Random forests, Neural Network, XGBoostand SVM on both the Ames dataset and Boston housing dataset and by finding optimal weights gain MSE of 0.0126. But the optimization they reach is specific to the set of models they have chosen. Additionally, research also shows that ensemble scores can be improved by adding a residual regressor [20].

### Formal Problem Setup (T, E, P)

**T:** Task at hand is the prediction of house prices. For each house i, it has attributes vector that describes features like lot area, number of garages that may influence the final sale price. We propose to learn a predictor to estimate the sales price of houses with attributes vector .

|  |  |
| --- | --- |
| **E:** The basis of our data is the Ames Housing dataset. The dataset contains 80 features (excluding ID) and 1460 instances. We split 20% of the data to be our initial test cases, and we will perform 5-fold cross-validation in our later tests.  **P:** We defined mean squared error as part of our performance evaluation metrics. RMSE is calculated as .  We use and compare the results for multiple linear and nonlinear regression models as RMSE shares the same unit as our y value. We compare the results of the base linear models with an ensemble of these models through stacking. Methods |  |

The dataset contains 80 features so we first used a correlation heatmap to explore the correlation between various features and the sale price. The heatmap shows that about half of the features has a correlation coefficient above 0.25. To avoid overfitting a dataset that has a higher dimensional feature space, we aim to select models with feature selection capacities.

We started by training a vanilla linear regression to be the basis of comparison. The small training loss and large test loss suggested the model may be overfitting. We then explored different machine learning algorithms to resolve the overfitting problem and minimize the loss function.

**Baseline approaches we compare against:**

Our baseline model is a linear regression model without regularization. It has a training RMSE of 21141 and a testing RMSE of 190212133572100. The baseline model having large test RMSE and variations in regressor coefficients suggest that the complex model for the limited amount of data is overfitting. Thus, we need to reduce dimensions by trimming irrelevant features to improve the model’s generalization performance.

**Implementation Details:**

***Linear Models:*** We first employed linear models to start from a simpler hypothesis space and to limit model complexity. To enhance the performance of linear models, we improved on the vanilla linear regression by adding regularization terms, applying log transformation to house sale price, and reducing dataset dimensionality with PCA.

***Other Models:*** In order to achieve a higher prediction accuracy, we ran 4 other types of models besides linear models - Support Vector Regression, Random Forest Regression, MLP Regression and Gradient Boosting Regression. These models were run by using the sklearn library.

For each model, an initial training run of the models on the full dataset were performed by using the default library hyperparameters.

Then to improve performance, hyperparameters were tuned by performing a grid search, utilizing the library’s GridSearchCV function. 5 fold cross validation was utilised during the grid search and the parameter space was selected manually. The hyperparameters that were tuned and the full list of values tested, can be found in Appendix 1. In the cases of Gradient Boosting and SVR, where the initial grid search yielded hyperparameters that were located at the extremes of the parameter space, a second grid search was performed on an expanded parameter space.

After the hyperparameters were tuned, the best parameters were chosen and models ran again with the improved parameters. After the hyperparameters were tuned, the best parameters were chosen and models ran again with the improved parameters.

***Ensemble model:*** In order to improve the performance of our predictions, our final step was to create a stacking ensemble model. To combine our previous models, a new train and test sets were created by taking the predictions of the previous models on the train and test sets and combining them into a single dataframe - each model constituting a feature in the new data. Then the data was standardized. We chose Gradient Boosting as our meta model, as it had the highest accuracy out of all the models that we ran. We first ran a model that combined Linear, Ridge, Lasso, PCA tuned Linear, SVR, Random Forest, MLP and Gradient Boosting regressions. This model performed on par with random forest regression alone and was not able to improve the prediction accuracy. It also didn’t benefit much from parameter tuning. This may be due to overfitting, as this model had the lowest train RMSE out of any model tested. To solve the problem of overfitting, we chose to remove some features through trial and error. The model that obtained the lowest RMSE after cross validations was the one where PCA tuned Linear regression, SVR and MLP Regression were removed from the initial ensemble set of features.

**Classification model:** The classification models try to address a different problem than exact price prediction. Policymakers might be more concerned about whether a house is an affordable house or a luxury house. In this project, we define houses with price below 25% quantile as low-end houses (encoded with 1), houses with price between 25% to 75% encoded with 2, houses with price higher than 75% encoded with 3. We apply SVM, with different kernels, and Random Forest for classification.

### Experimental Results:

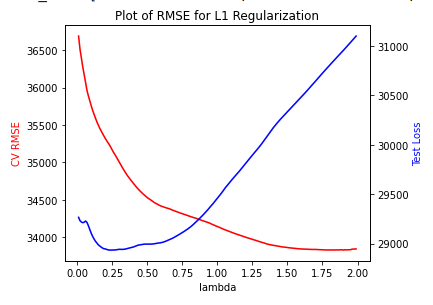
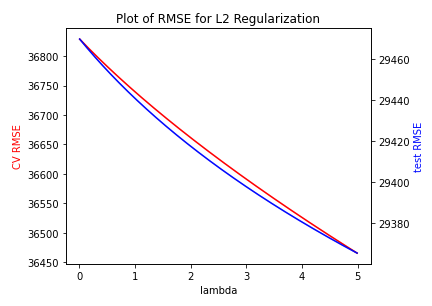
**Questions:**

We aim to answer the following questions: (1) What variables are significant in predicting house price and how do we obtain them? (2) What methods provide the lowest test error for linear models? (3) How do other models compare to linear models for house price prediction? (4) Whether we can use ensemble of models to achieve higher accuracy predictions for house prices? (5) Which classification algorithm we should use for highest accuracy?

**Linear models:**

We first observed that house price is positively skewed, so we proposed an improvement on the vanilla linear regression by applying a log transformation of house sale price to satisfy the normality assumption. In comparison to the baseline model, the large test RMSE signals that this model is still overfitting.

We proposed adding regularization terms to reduce model complexity, w. When tuning lambda in Ridge regression with 5-fold CV, the averaged CV RMSE keeps decreasing with larger lambda, suggesting that the model is still overfitting. When tuning lambda for Lasso regression, a model that encourages sparsity, we were able to find the optimal lambda parameter.



Then, we applied PCA to reduce dimensionality in the original dataset to address the overfitting problem. Preserving 80% of the total variance, PCA decreased feature dimensions from 243 to 97 columns. When fitting linear regression on PCA transformed dataset, test RMSE was significantly reduced. We also found that transforming y value also improved test performance. This result suggested that the original data contain many irrelevant features that do not have strong predictive power, so PCA is a useful tool to preserve structure relevant to predicting house sale prices. This approach answers question (1) and (2).

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Training RMSE** | **Test RMSE** |
| Linear Regression | 21141.58 | 190212133572100 |
| Linear Regression with L2 Regularization (lambda =4.99) | 21159.48 | 29365.55 |
| Linear Regression with L1 Regularization ((lambda =0.25) | 21141.57 | 29346.49 |
| Linear Regression with Log Transformation | 197183.07 | 47337528 |
| PCA Regression | 28818.64 | 36668.74 |
| PCA Regression with Log Transform | 24840.12 | 29331.92 |

**Other Models:**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Training RMSE** | **Test RMSE** |
| Support Vector Regression (Baseline) | 78941.42 | 88647.41 |
| Support Vector Regression (Tuned) | 28500.22 | 32979.62 |
| Random Forests Regression (Baseline) | 11106.91 | 29082.03 |
| Random Forests Regression (Tuned) | 13766.40 | 28821.57 |
| MLP Regression (Baseline) | 191151.08 | 193020.42 |
| MLP Regression (Tuned) | 7665.72 | 30091.99 |
| Gradient Boosting Regression (Baseline) | 13252.97 | 27343.64 |
| Gradient Boosting Regression (Tuned) | 11074.99 | 26755.02 |

All models experienced improvement in performance from tuning the parameters, with SVR and MLP improving significantly on the test set, while improvements for Gradient Boosting and Random Forests were more minor. Random Forests and Gradient Boosting achieve better accuracy on the test set than the best results achieved by linear models, while SVR and MLP were able to outperform only the basic linear model, indicating that their use for the problem at hand might not be preferred. This conclusion is also supported by the fact that the ensemble model performance was improved when these models were removed from the feature set.

**Ensemble model:**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Training RMSE** | **Test RMSE** |
| Ensemble of all tuned models (GBR, MLP, RF, SVR, PCALin, L1Lin, L2Lin, Lin) (Baseline) | 3445.94 | 28770.69 |
| Ensemble of all tuned models (GBR, MLP, RF, SVR, PCALin, L1Lin, L2Lin, Lin) (Tuned) | 2662.32 | 28702.32 |
| Ensemble of tuned models (GBR, RF, L1Lin, L2Lin, Lin) (Baseline) | 7043.39 | 24663.66 |
| Ensemble of tuned models (GBR, RF, L1Lin, L2Lin, Lin) (Tuned) | 5949.31 | 24644.66 |

**Classification models:**

|  |  |
| --- | --- |
| Method Name | Testing Accuracy |
| SVM - Linear Kernel | 0.8424657534246576 |
| SVM - Polynomial Kernel | 0.5 |
| SVM - Radial Basis Kernel | 0.4897260273972603 |
| SVM Sigmoid Kernel | 0.4383561643835616 |
| Random Forest | 0.8424657534246576 |

### Conclusions and Future Work

With linear models, we found that feature selection methods like PCA and Lasso regression which encourages sparsity enhance the generalization performance by reducing overfitting and model complexity. With ensemble models, we achieved the lowest test errors in testing by tuning hyperparameters, which decreased the test error by 15.98% over our best linear model. However, our model with the best performance still has limited usage as the average house value in our test set was approximately $180,000 and an RMSE of $24,644 suggests that our prediction has a 13.6% error rate on average compared to the actual sale price.

Further work on the ensemble model could involve selecting hyperparameters from a distribution in gridsearch in order to find even better hyperparameter combinations, as well as finding different combinations of models that could improve model diversity. For classification model, it might be more useful for policymakers and urban planners to change labeling strategy based on demographic information, and it could be more useful if the income of the residents could be incorporated in the data.

### Ethical Considerations and Broader Impacts:

While real estate evaluation using machine learning promotes pricing transparency and democratizes valuations for those lacking access to real estate agents, we should be mindful that this ML system could be flawed. Proxy features as input to the algorithm may lead to gentrification of neighborhoods that have been historically mispriced due to system injustice.

As the pricing of real estates has direct impacts on city’s tax planning, which in turn determines budgets for neighborhoods’ education and infrastructure, this ML model should only be seen as a facilitating tool rather than as an alternative to experts judgement.

### ---------- everything below can be over the 5-pg limit -----------------------

### Prior Work / References:

1. De Cock, Dean. "Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project." *Journal of Statistics Education* 19.3 (2011).
2. Kaggle House Prices - Advanced Regression Techniques challenge: <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview>
3. We first found the challenge through this affiliated dataset page on Kaggle: <https://www.kaggle.com/ngee379k/house-prices-advanced-regression-techniques>

*4 - 18 Kaggle notebooks that we reviewed*

1. <https://www.kaggle.com/maticortes/pipelines-iowa-data>
2. <https://www.kaggle.com/pdwivedi08/1st-machine-learning-model-submission>
3. <https://www.kaggle.com/saitharun97/advanced-house-price-prediction-part-3>
4. <https://www.kaggle.com/azizakinsola/house-price-predictor>
5. <https://www.kaggle.com/iverson3/house-price>
6. <https://www.kaggle.com/vishalakarnia/kaggle-competition-house-prices-regression>
7. <https://www.kaggle.com/rajivjoarder/house-prices-advanced-regression-techniques>
8. <https://www.kaggle.com/malathyshyamnair/advanced-regression>
9. <https://www.kaggle.com/niteshx2/top-50-beginners-stacking-lgb-xgb>
10. <https://www.kaggle.com/josh24990/simple-stacking-approach-top-12-score>
11. <https://www.kaggle.com/orhankaramancode/ensemble-stacked-regressors-top-3-92-acc>
12. <https://www.kaggle.com/satishgunjal/ensemble-learning-bagging-boosting-stacking>
13. <https://www.kaggle.com/eliotbarr/stacking-starter>
14. <https://www.kaggle.com/agodwinp/stacking-house-prices-walkthrough-to-top-5>
15. <https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard>

*19 - 20 Articles*

1. Shahhosseini M., Hu G., Pham H. (2020) Optimizing Ensemble Weights for Machine Learning Models: A Case Study for Housing Price Prediction. In: Yang H., Qiu R., Chen W. (eds) Smart Service Systems, Operations Management, and Analytics. INFORMS-CSS 2019. Springer Proceedings in Business and Economics. Springer, Cham.
2. P. A. Viktorovich, P. V. Aleksandrovich, K. I. Leopoldovich and P. I. Vasilevna, "Predicting Sales Prices of the Houses Using Regression Methods of Machine Learning," *2018 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC)*, Vladivostok, Russia, 2018, pp. 1-5.

### Supplementary material:

Appendix 1:

|  |  |  |
| --- | --- | --- |
| Model | Parameter space | Best Parameters |
| SVR run 1 | { 'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],  'C': [3.5, 4, 5, 10, 15, 20],  'epsilon': [0.0009, 0.0008, 0.0007, 0.0001, 0.00001],  'tol': [0.002, 0.00175, 0.0015, 0.00125, 0.001]  } | Best parameters found: {'C': 20, 'epsilon': 0.0009, 'kernel': 'linear', 'tol': 0.00175} |
| SVR run 2 | {  'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],  'C': [20, 40, 60, 100, 140, 200],  'epsilon': [0.1, 0.01, 0.005, 0.0025, 0.001, 0.0009],  'tol': [0.002, 0.00185, 0.00175, 0.00165, 0.0015]  } | Best parameters found: {'C': 100, 'epsilon': 0.001, 'kernel': 'linear', 'tol': 0.0015} |
| RFR | {  'n\_estimators': [50, 75, 100, 125, 150],  'max\_depth': [4, 5, 6, 10, 7, 8],  'min\_samples\_split': [3, 4, 5], # [2, 3, 4, 10, 100],  'bootstrap': [True, False]  } | Best parameters found: {'bootstrap': True, 'max\_depth': 10, 'min\_samples\_split': 5, 'n\_estimators': 75} |
| MLPR | {  'hidden\_layer\_sizes': [(50,50,50), (50,100,50), (100,)],  'activation': ['identity', 'logistic', 'tanh', 'relu'],  'solver': ['adam'], #['sgd', 'adam'], - sgd throws errors / maybe mention in the repport - lbfgs didnot converge  'alpha': [0.0001, 0.001, 0.01],  'learning\_rate': ['invscaling', 'constant','adaptive'],  'learning\_rate\_init': [0.001, 0.005, 0.01]  } | Best parameters found: {'activation': 'relu', 'alpha': 0.001, 'hidden\_layer\_sizes': (50, 100, 50), 'learning\_rate': 'invscaling', 'learning\_rate\_init': 0.01, 'solver': 'adam'} |
| GBR run 1 | {  'loss': ['squared\_error', 'ls', 'lad', 'huber', 'quantile'],  'learning\_rate': [0.001, 0.01, 0.1, 0.2, 0.8],  'n\_estimators': [10, 25, 50, 100, 200],  'max\_depth': [2, 3, 5, 7],  'validation\_fraction': [0.1, 0.15, 0.2]  } | Best parameters found: {'learning\_rate': 0.1, 'loss': 'huber', 'max\_depth': 3, 'n\_estimators': 200, 'validation\_fraction': 0.15} |
| GBR run 2 | {  'loss': ['squared\_error', 'lad', 'huber'],  'learning\_rate': [0.1],  'n\_estimators': [200, 220, 250, 270, 300],  'max\_depth': [3],  'validation\_fraction': [0.15]  } | Best parameters found: {'learning\_rate': 0.1, 'loss': 'huber', 'max\_depth': 3, 'n\_estimators': 270, 'validation\_fraction': 0.15} |
| GBR ensemble set 1, run 1 | {  'loss': ['squared\_error', 'lad', 'huber'],  'learning\_rate': [0.01, 0.1, 0.2, 0.8],  'n\_estimators': [50, 100, 200],  'max\_depth': [2, 3, 5, 7],  'validation\_fraction': [0.1, 0.15, 0.2]  } | Best parameters found: {'learning\_rate': 0.1, 'loss': 'huber', 'max\_depth': 3, 'n\_estimators': 200, 'validation\_fraction': 0.2} |
| GBR ensemble set 1, run 2 | {  'loss': ['lad', 'huber'],  'learning\_rate': [0.1, 0.11, 0.12],  'n\_estimators': [200, 220, 250, 270],  'max\_depth': [3],  'validation\_fraction': [0.15, 0.175, 0.2]  } | Best parameters found: {'learning\_rate': 0.12, 'loss': 'huber', 'max\_depth': 3, 'n\_estimators': 250, 'validation\_fraction': 0.2} |
| GBR ensemble set 1, run 3 | {  'loss': ['huber'],  'learning\_rate': [0.12, 0.14, 0.16, 0.18, 0.2],  'n\_estimators': [220, 250, 270, 280],  'max\_depth': [3],  'validation\_fraction': [0.2, 0.225, 0.25, 0.27, 0.3]  } | Best parameters found: {'learning\_rate': 0.12, 'loss': 'huber', 'max\_depth': 3, 'n\_estimators': 220, 'validation\_fraction': 0.2} |
| GBR ensemble set 2, run 1 | {  'loss': ['squared\_error', 'lad', 'huber'],  'learning\_rate': [0.01, 0.1, 0.2, 0.8],  'n\_estimators': [100, 200, 250, 270, 300],  'max\_depth': [2, 3, 5, 7],  'validation\_fraction': [0.1, 0.15, 0.2]  } | Best parameters found: {'learning\_rate': 0.1, 'loss': 'huber', 'max\_depth': 3, 'n\_estimators': 300, 'validation\_fraction': 0.1} |