Housing Prices - Gradient Boosting Techniques

Machine Learning CS 4347

Luka Brown & Sean Angle

Dr. Vangelis Metsis

5/2/23

Abstract

The Kaggle Housing Prices competition requires predicting the sale prices of houses based on a set of features. In this project, we employed several feature selection techniques, including Principal Component Analysis (PCA), Elastic Net, and Variance Threshold, to identify the most relevant features and improve model performance.

PCA was used to reduce the dimensionality of the data and extract the most significant features. Elastic Net was also employed, a regularization technique that combines the strengths of Lasso and Ridge regression, to select features that have a high impact on the target variable while preventing overfitting. Additionally, Variance Threshold was used to select features with high variance, which indicates that they have a higher impact on the target variable.

The highest scoring model achieved an R squared score of 0.887 on the test data, which placed the submission in the top 25 percent of the competition leaderboard. Different combinations and methods and strategies outlined in this document show our process and methods to achieve this score.

Overall, this project highlights the importance of preprocessing and feature selection techniques in building accurate predictive models. Our submission also demonstrates the effectiveness of standard scaling, one hot encoding, and variance filtering in improving and tuning model performance.

Project link: https://www.kaggle.com/code/lukabrown/gradient-boosting

1. Introduction

The Housing Prices competition on Kaggle is a popular machine learning challenge that aims to predict the sale prices of houses based on various features such as location, size, and quality. The problem is of great practical importance in the real estate industry as accurate predictions of house prices can help buyers and sellers make informed decisions.

The challenge requires participants to clean and preprocess the given dataset, which includes over 70 features and 1,400 observations, and build a predictive model that can accurately estimate the sale price of a house. The dataset also contains missing values and outliers, which adds complexity to the problem and requires advanced data preprocessing techniques.

Several approaches have been proposed to tackle this problem, including linear regression models, decision trees, and neural networks. Preprocessing methods include standard scaling, one hot encoding for the numeric and categorical features respectively have been used as well. A hybrid of median/mode imputing and feature dropping were used to fill in missing test data. Additionally, feature engineering techniques such as Principal Component Analysis (PCA), Elastic Net, and Variance Threshold have been employed to select the most relevant features and improve model performance.

Overall, the Housing Prices competition presents a challenging problem that requires a combination of data preprocessing, feature engineering, and model building skills. The competition has attracted a large community of data scientists and machine learning enthusiasts who continue to propose novel solutions and push the boundaries of predictive modeling in the real estate industry.

2. Methodology

The first step in making the data usable for learning using gradient boosting was to employ tactics to deal with the missing data. After initial research, it was shown that all 1459 entries in the test dataset had some form of missing data. Since imputing data is based on other real data being present in the feature, first the columns that were missing more than 50 percent of their data were selected to be removed. 5 features were removed with 4 out of 5 of those features missing greater than 80 percent of its data.

Remaining missing data to deal with included LotFrontage with a 15.5 percent missing. This data had many N/A values that represented that there was not a lot present. This was a numerical feature and just needed the values to be mapped to 0. This left a remaining 27 columns with less than 100 missing values each. Our project included multiple approaches and found that using KNN imputation on the values did not lead to a lower mean squared error. Median and mode imputation is a simpler approach but did lead to better results. The categorical features selected the mode of the column to fill the missing values. The mean, or mode, was used to fill in the numerical feature's missing values. After these features have been filled, the dataset is ready for further preprocessing and feature selection.

Preprocessing this dataset involves two main processes: one hot encoding and standard scaling. The scaler takes numerical data and evenly distributes the data around 0 which helps many machine learning algorithms perform better. One hot encoding is a way to deal with the categorical features by creating another column or columns that binarizes a single category. Through employing both of these methods on the dataset the features are expanded to 270. The dimensions are quite high so multiple methods of feature selection must be used to simplify the model while maintaining impactful features.

Variance threshold was the first method used and removed a significant amount of features that had a variance lower than 0.1. 187 features were removed and most of those were new one-hot columns added. Others were the numerical features being removed, leaving the dataset with 83 features. The features dropped were very unlikely to contribute much to getting a lower mean squared error. In further testing this was shown to be true, as our model will perform better after this step. There were still many features and potentially some that could still be removed in another process.

Using an elastic net, the model was fit to our training dataset to determine if the coefficients would be able to provide insight on the correlation of some features. Through extracting and reviewing the data, a threshold of 100 was established and any coefficients of features that had an absolute value of greater than 100 were to be removed. Depending on the run, this method only removes around 2-5 features. At this point about 70 percent of features of the original 270 have been removed. This indicated that our selected features have a high correlation with the target variable and this led us to our last method.

80 dimensions is still very high but there is a way to help this: dimensionality reduction. Principal component analysis (PCA), works using complex linear algebra to project the data downwards into lower dimensions while maintaining as much of the info and relationships as possible. This required a bit of testing to determine what the best number of dimensions to reduce to was and through many runs it was

determined that 40 dimensions yielded very good results for this dataset. This concluded our feature selection process and now our dataset is ready to be fit onto the gradient boosting model.

The model selected has multiple variables that can be customized depending on the dataset. A grid search model allows for multiple variable combinations to be assessed against one another very easily. Our parameters for the model included n_estimators, max_depth, and learning_rate. The n_estimators is the number of boosting stages to perform. Sklearn states, “Gradient boosting is fairly robust to overfitting so a large number usually results in better performance [1].” Knowing this, values ranging from 100-250 in intervals of 50 were chosen. The max_depth indicates how deep the decision tree will go, and a range from 2-4 worked to give us our best results. Lastly, the learning_rate was set to 0.1 as that yielded the best results.

The parameter grid will allow the model to determine what is best between each run to give us the best results. Mean squared error is the scoring metric chosen because the challenge is graded off of that same metric. This grid search takes a couple of minutes to run depending on which machine it is running on. The model was fit to our post-PCA dataset and then used to predict the output of the test dataset. The test dataset is split from the training dataset provided by Kaggle and it was obtained using a 20 percent split. This allows us to evaluate our models using metrics other than mean squared error. R squared is another metric used in this project to determine if the predictions are better or worse.

Lastly, the competition required that the submission be output to a csv file, so the last cell of the project is just output. The ID needs to be manipulated to reflect the original test dataset ID’s, which were dropped in preprocessing, so that is completed at this time with an offset of 1461. After putting in the index of the output column, the csv is ready for the pandas .to_csv to complete the process. The submission contains 1459 entries of the predictions made by the model, and our top scoring submission on Kaggle beats ~75 percent of submissions.

2.1 References

1. *Gradient Boosting Regressor*. Scikit Learn, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html.
2. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

3. Results

Even though PCA and the elastic net were written out, the model performed better on the dataset after filtering for the variance only. This is reflected in version 18 of the Kaggle notebook, the best scoring submission, where PCA and the elastic net are commented out. This means that those methods of feature selection were not the most optimal for this dataset, or simply that many of the features had very high correlation already. The best parameter grid still involved n_estimators of 100-250, and a max_depth of 2, 3, and 4. Slight improvements were also made in the train test split line, where it was changed to only 1 percent of training data will be split into the test for evaluation. It was not necessary for a large test dataset as most of the tuning of the model had been completed, and 1 percent was enough to give an idea of the mean squared error.

The initial submission placed our notebook in the top 45% of scorers but that left a lot of room for improvement. At this point began model tuning and began evaluating and testing different parameters. First to be tuned was the number of dimensionals to reduce down to while utilizing PCA. After many runs, typically in increments of 5, 40 was yielding the best results and was used from then on. Next to be tuned was the parameter grid for the GridSearchCV. It was quickly determined that the model was selecting a learning rate of 0.1 out of our initial range of 0.01 and 0.1. This left n_estimators and max_depth up next. N_estimators was initially set to 100, 200, 300, but after testing in increments of 50, the sweet spot was shown to be 100, 150 and 200. Similarly, the max_depth sweet spot was found to be 2, 3, and 4. Lastly, one last variable parameter was also tested: drop_first being true on the one hot encoding model. This would drop the first class found in each feature, but did not improve the gradient boosting model in any way so it was not selected.

Figure 3.1

submission.csv (34.47 kB)

Id	SalePrice
1461	111033.32309463993
1462	161341.9241315011
1463	183707.86553569703
1464	185027.96404282053
1465	192457.70147028414
1466	176864.2540248915
1467	171247.23856584547
1468	167043.87815456174
1469	189143.47770622562

Figure 3.2

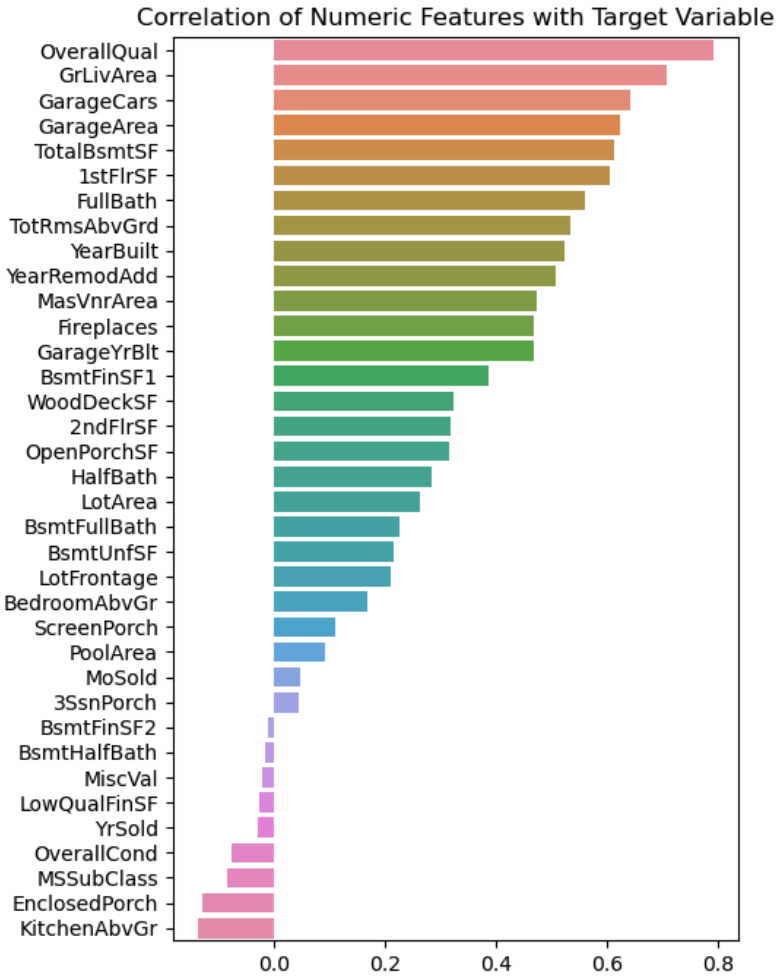

Figure 3.3

4. Conclusion

In this project, we employed various methods to preprocess and select features to prepare our dataset for gradient boosting, a powerful machine learning algorithm. We began by dealing with missing data, removing columns with missing data greater than 50 percent and imputing the remaining data using median and mode imputation. We then proceeded to preprocess our dataset using one-hot encoding and standard scaling, which led to an expansion in our dataset to 270 features. To reduce the number of dimensions, we employed variance threshold, elastic net, and principal component analysis to select the most impactful features.

Our final model was a gradient boosting model with n_estimators, max_depth, and learning_rate as the customizable parameters. We used a grid search model to assess various parameter combinations and selected the best model based on mean squared error, the scoring metric chosen for this project. We split our test dataset from the provided training dataset and evaluated our model using other metrics such as R squared to ensure the predictions were accurate.

Our top scoring submission on Kaggle was able to beat ~75 percent of submissions, demonstrating the effectiveness of our model. However, there are always areas for improvement. One potential area for improvement is to further optimize the parameter grid search and try more combinations to see if a better model can be obtained. Another would be to use ElasticNetCV, which would cross validate itself and perhaps provide more accurate values. Furthermore, another way to improve would be to use a different embedded method, such as simple linear regression to determine which coefficients to remove. Additionally, we can explore other feature selection techniques, such as recursive feature elimination, to further simplify the model and improve its performance.

In conclusion, this project gave us invaluable practice in feature selection, preprocessing, model optimization, and assessment. Future initiatives will be well-supported by the methods employed in this study since they may be utilized with a variety of datasets and machine learning algorithms. The learnings from this project may also be used for future optimizations and upgrades, serving as a standard for further performance assessments. Overall, this initiative has advanced the fields of data science and machine learning and shown how these technologies may be used to generate insightful data and influence future markets.