Dylan Dunda

CIS 3715: Principles of Data Science

Hongchang Gao

24 April, 2022

Predicting Housing Prices with Linear Regression

# Introduction

For my final project, I chose to put what I’ve learned in this class to the test by training and testing linear regression models on the housing prices dataset provided by Kaggle.com for the “House Prices - Advanced Regression Techniques” challenge. My motivation behind choosing this project was that after I read through it and figured out exactly what it would entail, it seemed like the perfect opportunity to put my skills to the test. While browsing through some possible projects, I wanted to make sure that I chose one that I would be comfortable with so that I could hopefully end up with some working code and decent results. A few of the other projects seemed to require some additional techniques and code that I felt could have been a bit overwhelming for me, but after looking through this assignment and the given dataset, I knew I had a lot of valuable resources from past classes/labs that would apply here. I also chose this topic because out of all of the projects, this one seemed the most concrete. To me, some of the other projects, like the idea of trying to predict exactly which passengers survived the crash on the Titanic seemed too abstract. I was not able to find any other projects that piqued my interest more than the housing prices project from Kaggle, and I knew going into the project that it would be a solid dataset, so I ended up going with it.

The goal of this project was to use the given training dataset from Kaggle.com in order to predicting the prices of houses. The dataset included details about a multitude of features for houses in Ames, Iowa. These features were meant to be used in order to predict the target variable, the hosing price, for other houses from the testing set. To accomplish this, I used a lasso regression model and a ridge regression model, then compared the results and chose the better of the two.

To achieve my goal of being able to predict the housing prices for houses in Ames, Iowa, I need to break the problem down into some steps. First, I needed to explore, analyze and preprocess the given data to ensure that is useful/applicable to the prediction model that I was going to use. This would include techniques like data cleaning and transformation. Next, I needed to implement and train the prediction models. It was important to use the same processed data for each model. After that, I had to evaluate the performance of each model. I did so by finding values that were useful in the comparison of prediction models such as the mean squared error, the mean absolute error, the explained variance score, and the r squared score. Lastly, I needed to compare the results of the two models to see which one was more effective.

# Approach

As mentioned in the introduction, the first thing I had to do for this project was to explore and analyze the given data from Kaggle.com.

To begin, I took a look through the dataset to see if there were any notable traits. There were a couple features that I suspected would have a relatively strong correlation with the housing price target variable, being “OverallQual”, “OverallCond”. “OverallQual” was the rating of the overall material and finish of the house. “OverallCond” was the rating for the overall condition of the house. I suspected that these features would have a strong correlation with the sale price because they were assessed based on the entire property rather than a single specific part or aspect of t. These features are also somewhat subjective. The scores for the overall quality or condition of the house may differ depending on who is judging it, but when it comes the assessing the data and training a prediction model, these categories can still be fairly valuable because high scores in these features show that someone is consciously assessing the overall property as “high quality” or “good condition.”

There were also a few features that I suspected would not be very valuable in predicting the sale price, such as “PavedDrive”, and “MoSold”. “PavedDrive” was a feature that contained information on if the house had a paved, partially paved, or dirt/gravel driveway. I did not see a paved vs. gravel driveway as something that would greatly affect the sale price of the house. Although I figured there would be some correlation between the house price and the driveway, being that the more expensive a house is, the more likely it is to have a paved driveway, I did not think the category would have a strong causative effect on the price. Although it may be true that the most expensive houses have paved driveways, there is no reason why such a house can not have a dirt/gravel driveway. For example, it is common for very expensive houses to be on private land, and if this land requires the use of a very long road/driveway, it might not make total sense for the whole thing to be paved. When it comes to the “MoSold” category, I did not suspect it to have a big effect on the house price target variable. In contrast, the year the house was sold would be valuable because this would take into account things like inflation of currency and the state of the housing market.

After studying the data a bit and learning about each of it’s features, it was time for me to preprocess the data. This included dropping irrelevant features, assessing outliers, handling missing values, transforming categorical values to numerical values, and normalizing skewed data.

I started preprocessing the data by dropping the “ID” column because it was completely unnecessary for the prediction process. I then dropped the “Utilities” feature because all but three records were the same value, so it did not contribute to the overall prediction model. Around this point, I also went ahead and deleted some major outliers.

The next step was to transform all of the categorical features into numerical features. In this dataset, there were 43 categorical features: MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConfig, LandSlope, Neighborhood, Condition1, Condition2, BldgType, HouseStyle, RoofStyle, RoofMat1, Exterior1st, Exterior2nd,MasVnrType, ExterQual, ExterCond, Foundation, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, Heating, HeatingQC, CentralAir, Electrical, KitchenQual, Functional, FireplaceQu, GarageType, GarageFinish, GarageQual, GarageCond, PavedDrive, PoolQC, Fence, MiscFeature, SaleType, and SaleCondition. I converted them by using label encoding. While label encoding was effective for transforming the categorical features to numerical features, ordinal encoding may have been more suitable for some features. In the future, if I were to redo a project similar to this, I would specifically choose ordinal encoding for the feature where it would be applicable, but I did not do so in this project.

After handling the categorical features, I continued by checking which features had missing values. In this dataset, there were 19 features that contained null values: LotFrontage, Alley, MasVnrType, MasVnrArea, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, Electrical, FirepaceQu, GarageType, GarageYrBlt, GarageFinish, GarageQual, GarageCond, PoolQC, Fence, and MiscFeature. In order to address these values, it was important to understand that features related to a garage will be NA if the house does not have a garage, and features related to a basement will be NA if the house does not have a basement. These features can’t be replaced with any calculated values, as they would be inaccurate and would negatively affect the prediction score. So when these features have missing values, it will almost certainly decrease the sale price of the house. Logically speaking, if two exact same houses are compared to each other, except one has a garage and the other doesn’t, the house with the garage will be more expensive. So the missing values can’t be filled in with values that correlate to the features of a garage when no garage exists for the house, because that will inaccurately increase the prediction for the sale price of the house. I dealt with the missing values with a few different methods depending on the feature. Some of the categories with missing values were best substituted with a median value, while others were better substituted by a zero value, ‘None’ value, or the most common value from the feature. I determined which method was best by reading the description of each feature with missing data and looking at the provided data. I knew I was complete with this step when there were no remaining missing values in the dataset.

I then moved on to addressing the skewed features. It was important to address the skewness of some of the data because linear regression models perform better with normalized data. So, I checked for skewed features, of which I found 49. I normalized these features using a Box Cox transformation, then moved on to “SalePrice” target variable. I analyzed the “SalePrice” variable distribution with a graph and came to the conclusion that it was right skewed. I corrected the skew by applying the numpy function log1p which applies log(1+x) to all of the elements in the feature. This can be seen with the visuals from before normalization (left) to after normalization (right).

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

This concluded the preprocessing phase of the project, so I moved on to implementing the models. For this particular project, I used two models, a lasso regression model and ridge regression model. I chose these models simply because I felt the most comfortable with them, and I also because I knew they would apply to the given data. But, after doing some research, I’ve seen a lot of other people get good results with different models, such as elastic net regression, gradient boosting regression, XGBoost, and LightGBM. If I were to continue to work on this project, it would be interesting to compare the results of all of these different models. I also found other solutions which included stacking multiple models to get a single result, and while I’m not entirely sure how this is implemented, it is another possible advancement that would be interesting to explore in the future if I were to continue to try to improve the overall accuracy in my project. But, for the sake of this project, my goal was to compare the results between my two models, lasso regression and ridge regression, and see which one was more effective for this dataset.

I began with the lasso regression model. In order to implement this model, I used sklearn’s linear model lasso regression. I also used sklearn’s Robustscaler() function in order to mitigate the effects of the lasso regression’s sensitivity to outliers. I then continued by implementing the ridge regression model. To implement this model, I used sklearn’s kernel\_ridge regression. For both of the models, I used the cross validation score function which came from sklearn as well.

# Results

After training the models, I evaluated them based on four scores: the mean squared error, the mean absolute error, the explained variance score, and the r squared score. I also recorded the standard deviation for each of these scores to ensure they were somewhat reputable.

The mean squared error and mean absolute error are both measurements to see the difference between the predictions and the true values. For these scores, the best possible score would be zero, meaning there was a zero percent error, and every prediction was the same as the true value. This would be the score for an absolutely perfect prediction model. While that is not entirely feasible for a project like this, I did manage to get some scores that showed that my model was somewhat accurate. The lasso regression model earned a mean squared error score of 0.1195 and a mean absolute error score of 0.2914, with standard deviations of 0.0052 and 0.0021, respectively. The ridge regression model earned a mean squared error of 0.1664 and a mean absolute error of 0.3424, with standard deviations of 0.0076 and 0.0092 respectively. In this regard, it’s clear that the lasso regression model had the better scores.

The explained variance score is used to measure the discrepancy between a model and actual data based on the total variance. The optimal score here would be 1. With my implementation, the lasso regression model earned an explained variance score of 0.9541 with a standard deviation of 0.0036. The ridge regression model earned an explained variance score of 0.9087 with a standard deviation of 0.0102. Based on these results, it is clear that the lasso regression model outperformed the ridge regression model.

The r squared score, or the coefficient of determination, is the measure of how close the data is fitted to the predicted regression line. The optimal score here is 1. In my project, the lasso regression model earned an r squared score of 0.9539 with a standard deviation of 0.0036. The ridge regression model earned an r squared score of 0.9083 with a standard deviation of 0.0103. With these results, it is apparent that the lasso regression model performed better than the ridge regression model.

# Conclusion

After taking a look at all of the results, the lasso regression model clearly outperformed the ridge regression model in all aspects. This makes sense because of the differences between the lasso regression model and ridge regression model. While the coefficients of the linear transformation are normally distributed in the ridge regression model, they are Laplace distributed in the lasso regression model. This makes it easier for the lasso regression model to have coefficients of zero, meaning it is easier to eliminate some of the features that do not largely affect the target variable. With such a large number of features in this dataset, it is to be expected that some of the features do not contribute enough to be considered in the prediction of the target variable.

Overall, I am happy with the outcome of this project. I was able to successfully implement the two models and get decent scores for each. There is a pretty clear difference between the two, with the lasso regression beating out the ridge regression to be the most accurate.

# Acknowledgements

The most impactful resource I had for this project was this tutorial on Kaggle.com: <https://www.kaggle.com/code/serigne/stacked-regressions-top-4-on-leaderboard/notebook>

While this particular implementation used stacked regression to get an optimal result, it was still very helpful because it went into depth about how to deal with each specific feature for data preprocessing and was overall just a great page filled with valuable information.

Another Kaggle.com notebook that proved to be very helpful was this one: <https://www.kaggle.com/code/pmarcelino/comprehensive-data-exploration-with-python>

The author of this resource, Pedro Marcelino, does a great job of explaining data science techniques in layman’s terms. It really makes some topics a lot easier to understand when they are presented in this fashion, and it helped me to wrap my head around exactly why I was doing certain things in this project and how they affected the results.

The labs for this class were also incredibly useful to me for this project. Taking on a large project like this can be a bit overwhelming, but since almost all of the topics that I covered in the labs proved to be applicable for this project, it helped me feel a bit more comfortable when implementing code because I was already familiar with it.