STAT 542, Project 1 Report

Jean Liu, Shivesh Pathak, Ashley Hong, Paulina Koutsaki

March 1, 2017

1 Cleaning and imputing the data

Imputing measurements of categorical predictors In the raw data there are many instances of the measurement NA in both categorical and numerical predictors. To move forward with any kind of learning we need to first understand the meaning of these NA measurements and possibly alter their values to match our understanding. By reading the provided data description on Kaggle, we find that NA values for the categorical measurements are not missing, but rather correspond to specific factors themselves. For example a measurement of NA for the predictor Fence does not mean that the measurement is missing, but rather that there is no fence at all! Therefore to treat this problem properly one would need to take all of the NA measurements and map them to a factor. In the case of Fence, NA becomes ”No Fence” for example. We do this kind of mapping for all categorical variables that have NA measurements as follows: PoolQC, NA becomes ”NoPool” if PoolArea=0 for the same measurement; MiscFeature, NA becomes ”None” if MiscVal=NA or 0 for the same measurement; Alley, NA becomes ”NoAccess”; Fence, NA becomes ”NoFence”; FireplaceQu, NA becomes ”NoFirePlace” if Fireplaces=NA or 0; GarageYrBuilt, GarageFinish, GarageQaul, GarageCond, and GarageType, NA becomes ”NoGarage” if GarageArea=NA or 0; GarageArea, GarageCars, NA becomes 0; BsmtFullBath, BsmtHalfBath, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, NA becomes 0 if TotalBsmtSF=NA or 0; BsmtQaul, BsmtFinType1, BsmtFinType2, BsmtExposure BsmtCond, NA becomes ”NoBasement” if TotalBsmtSF=NA or 0.

Advanced cleaning of categorical predictors After this naive cleaning we get reasonable results when learning is done, but we found that applying some additional cleaning of the data actually improves the accuracy of our models. Additional cleaning takes into account that some of the categorical variables are related. For example, MasVnrType and MasVnrArea are faulty since measurements of NA in MasVnrType many times have positive MasVnrArea. To remedy this, every NA measurement in MasVnrType becomes ”None” and the corresponding MasVnrArea is set to zero. For MSZoning we take the missing values to be ”RL”. For BsmtFinType2 we set NA to ”ALQ”, for BsmtQual we set NA to "Gd" if HouseStyle is "2Story" and "TA" if HouseStyle is "1.5Fin", and BsmtCondition to "TA." The reasoning for these last four choices can be found in an excellent kernel by Phillipe Gaudreau.[1]

Imputing numerical predictors There is only one numerical predictor that has missing values, LotFrontage. Phillipe's analysis shows that there is a way to predict the missing LotFrontage data using the training data we have, but we found it to be too tedious and not very helpful. Instead, we take inspiration from the work of Kaggle user meikegw's [2] analysis between LotFrontage and LotArea and simply set LotFrontage to be the square root of LotArea when the measurement of LotFrontage is NA.

New levels When we split the training and test data there are some case where we have new levels or just single factors with non-zero frequency. To deal with this I simply merge the measurements in the new level into the most frequent level of that factor.

2 Simple linear regression

To start off the analysis for Iowa housing data, we decided to first implement a simple model as a benchmark. Depending on how our linear model fits, we will try to make a few more complex models to have a better prediction. For the simple model, we decided to fit a multiple linear regression to our data with all the given variables. All the variables seemed important in terms of predicting housing price, so we didn't intentionally drop any of them (except for Utilities and Electricity as above). After fitting multiple linear regression to Iowa housing data set, we saw which variables were significant in predicting the housing price. The adjusted R square is around 0.93. The calculated RMSE is usually around 0.13, which can be used for comparing our models. The simple linear regression runtime was roughly 0.15s, which is very quick!

3 Ridge/Lasso regression

Next we tried variable selection with AIC/BIC as well as Ridge and Lasso to see if they could produce better predictions. We first tried stepwise variable selection to predict sale prices, with forward selection as well as backward selection, but the RMSE was even worse than that of the simple linear method, so we decided to give up on stepwise.

Then we turned to Ridge and Lasso, where all the qualitative variables were turned into dummy variables, and the dataset was split into training and testing data. We used the glmnet package to perform ridge and lasso, λ was defined between 10^{-2} and 10^{10}. The best value of lambda that would minimize the error was chosen, in this case, cv.glmnet was applied to find out the minimum

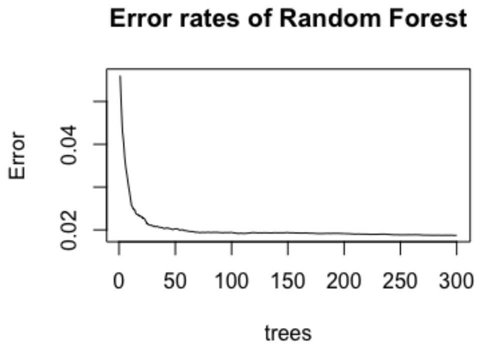

Figure 1: Error rates of Random Forest against the number of trees

error, in which procedure the two regressions bridges the gap differently, lasso ($\alpha = 1$, the default) and ridge ($\alpha = 0$). The models with best lambda were tested: for ridge this produced an RMSE of approximately 0.013, and for lasso an RMSE of around 0.015. However, for some samples the lasso outperforms the ridge method, and hence we choose the model that provides a lower RMSE. The Ridge method takes roughly 1.70s and the Lasso takes roughly 3.45s. The Lasso procedure takes about 20 times longer than the simple linear regression and the Ridge takes 10 times longer.

4 Random forest regression

To capture possible nonlinearity in the data we grew a random forest. Our forest consists of 200 trees, since this is where the out-of-bag error stabilizes, as shown in Figure 1. Lastly, by using *tunerf* to tune the number of attributes m, we reached the conclusion that the default value $\lfloor p/3 \rfloor$, 26 in our case, works well for this data set, and was therefore left unchanged. The RMSE for the random forest method is typically lower than all of the prior methods, around 0.011, but at the cost of a lot more computational time. The random forest method usually takes 21.5s, which is significantly longer in time than the simple linear method, about 150 times longer.

5 References

1. https://www.kaggle.com/clustersrus/house-prices-advanced-regression-techniques/house-prices-dealing-with-the-missing-data/notebook
2. https://www.kaggle.com/meikegw/house-prices-advanced-regression-techniques/filling-up-missing-values