Report of Predict the Housing Prices in Ames

Data Pre-processing

1 Remove Outliers

According to data document, several observations with GrLivArea > 4000 are outliers that are either true outliers or very usual sales. So we remove these observations from train dataset.

2 Missing Data Imputation

a Categorical Variables

According to data document, there are two types of missing data. For some categorical variables, e.g. "Alley", "BsmtQual","BsmtCond", the NA’s actually mean that the house don’t have these features. Thus, we code missing in these variables into a new level of factor: “None”. On the other hand, some categorical variable, e.g. “MasVnrType”, already has a level called “None”. In this case, the NA’s are true missing values, so we substitute them with the most frequent level of the factor.

b Numeric Variables

For numeric variables, we replace any missing value with the mean of that variable.

3 Data Transformation

We apply log transformation to the dependent variable SalePrice.

Although we tried applying log transformation to all numeric variables with skewness > 0.75, it did not work well and made prediction worse, so we decided not to log-transform any independent variables in the end.

Models

1 Simple Linear Regression

SalePrice ~ MSZoning, LotArea, OverallQual, OverallCond, YearBuilt, BsmtFinSF1, TotalBsmtSF, GrLivArea, BsmtFullBath, KitchenAbvGr, KitchenQual, Fireplaces, YearRemodAdd, GarageCars, ScreenPorch, SaleCondition

We include 16 variables to fit a linear regression model. These variables are selected by stepwise BIC. We perform stepwise BIC on five different subsets of train data and choose the most common variables as our predictors in simple linear regression.

2 Bagging LASSO

We combine LASSO and bagging technique in this model. Each time we sample randomly from train data, fit a LASSO model to this new subset of train data, and make predictions on test data. We do this process for 200 times, and take the average of predictions as our final prediction. In each LASSO model, we chose lambda with smallest standard error as the best lambda by 10-fold CV.

3 Xgboost

We tune our xgboost model as follows: First, we code levels of factors into integers. Second, we find the best rounds and best max depth by 5-fold CV. After 1500 rounds, the performance of the model changes very little while runtime increasing. So, we choose the best rounds to be 1500. And the best max depth is found to be 12, beyond which it starts to overfit the data and accuracy drops.

Model Comparison Based on Run Time and Accuracy

		1st Test	2nd Test	3rd Test	4th Test	5th Test	**Mean**	**SD**
Linear Model	**runtime**	0.04	0.05	0.05	0.01	0.04	**0.04**	
	rmse	0.1160	0.1189	0.1335	0.1178	0.1150	**0.1160**	**0.00758**
Bagging LASSO	**runtime**	35.13	30.83	33.09	28.80	28.79	**35.13**	
	rmse	0.1223	0.1230	0.1386	0.1262	0.1112	**0.1223**	**0.00979**
Xgboost	**runtime**	34.36	35.28	35.78	35.15	34.97	**34.36**	
	rmse	0.1724	0.1726	0.1780	0.1702	0.1451	**0.1724**	**0.01293**

The table above shows that the average run-time of linear model, which is 0.04, is far less than those of LASSO and Xgboost, which are both almost 35. When it comes to accuracy, linear model has the least rmse which is 0.1160 with least standard deviation while Xgboost has the largest rmse which is 0.1724 with largest deviation. In conclusion, simple linear regression is the most efficient and accurate. Bagging LASSO is slightly less accurate than linear regression despite of longer runtime. Xgboost is the least accurate model. However, it is worth to mention that bagging LASSO and Xgboost are much more robust than simple linear regression.

Discarded model: PCA

We tried PCA to reduce the dimensions. It turned out that the first component explained more than 98% variance and the first two components explained more than 99% variance. This deducted dimension too much. Thus, we discarded this method.