Draft report

2022-11-27

Data Pre-processing

Our dataset is the Ames Housing dataset compiled by Dean De Cock for use in data science education. These data were gathered from Ames Iowa during a period spanning 2006 to 2010. The observational unit is a house sold in Ames from 2006 to 2010. The response variable, house sale price, is quantitative. The dataset includes 79 predictor features, with more than half of our predictors categorical. The predictors range from geographical information about the property and the neighborhood, to interior measurements and quality of amenities, to sales statistics. The dataset has 1460 observations. We found this dataset through the "House Prices - Advanced Regression Techniques" Kaggle competition, available here.

Several important steps were taken in order to prepare this dataframe for our statistical analysis. The first step was recoding NA's. In the raw .csv file, many of the categorical variables took the value NA frequently. Upon reading the codebook, it became clear that in many of the variables, the value NA actually stored meaning. Hence, whenever NA was listed as a possible value storing meaning for a categorical variable in the codebook, we replaced the value NA with the character value "None" so that our algorithms would not treat these NAs as missing data. Next, the continuous variable `LotFrontage` contained a large number of NA values. This variable records the length of street/sidewalk bordering a home. After some exploratory analysis, it became clear that instead of recording "no access" as 0, it was recorded as an NA. Hence, we recoded NA as 0 in LotFrontage. In any case where the codebook indicated that a variable which took numerical values should be interpreted as categorical (variables such as house legal subclass and month sold), we changed the variable type to character.

The next issue to be addressed was rare levels. Some of the categorical features contained levels that were exceedingly rare. In extreme cases, levels were observed only once or twice. This imbalance in levels created issues in data splitting both for cross validation and for the validation set. Since the problem was so acute, even stratified sampling was not an option. Our solution was to combine the rare levels into a single level "other." We set a threshold of 10 observations in a level. Levels with fewer than ten observations were collapsed into the single "other" category. This solution was designed to balance functionality with the desire to preserve the original data structure. Ten observations in each level gives an expected number of 2.5 observations in each level in the test set and 7.5 in the training set, which means that we likely will have at least one observation from each level in the training set and test set. When all levels within the test set are contained in the training set, our models will be able to make predictions on the test data, which will enable us to draw conclusions about relative model performance.

Even when combining rare levels into one "other" category, there were still categorical variables in which the other variable had less than 10 observations. These rare instances created the same issues with CV and in the training test split, so it was decided that these rare "other" values would be overwritten with the mode of the categorical variable. These rare observations reflect less than 0.68% of the observations in each level, so the alteration to the data from imputing was not at all substantial. To prevent data leakage, when we imputed the mode, we imputed the training mode for training observations and the test mode for test observations. If imputation created a variable that only took one single value, then that variable was removed from the whole dataset. This occurred for the variables `Street`, `Utilities`, and `PoolQC`.

After handling the rare values, there were still some variables which had to be cleaned up. The variable `GarageYrBlt` records the year that the home's garage was built. If the home did not have a garage, this value was stored as NA. There were 81 of these values which is about 5% of the overall rows. For these NA's, we imputed the mean (for the test and training sets), as not to have these rows discarded. Next, the variables

recording Masonry Veneer Type and Masonry Veneer Area had 8 missing values for the same 8 rows. The mode of "none" (across both sets!) was imputed for Veneer Type and the value corresponding to "none," 0, was imputed for the Veneer Area. Finally the one remaining categorical variable which truly did take NA values (the NA only stored the information that the data were missing), `Electrical`, took only one NA value and the mode value of the set it landed in was imputed here.

One specter looming over all of this data processing is cluster analysis. A more theoretically justified and effective way to deal with the rare levels phenomenon and collapse the dimensions on our dataframe could be PCA for mixed data (or an analogous decomposition through cluster analysis). By isolating principal components, many of the issues with high dimensionality and rare levels would be resolved and our models would likely be more effective. Unfortunately, we have yet to reach these methods in class, and the authors haven't had too much time outside of class to learn about these methods. The "other" recoding method was the most effective strategy we could come up with using the tools that we have learned about in class. If we learn about cluster analysis and PCA in time for the final presentation, we will test the best performing method on the processed data (as uncovered in this report) on principal components to see if it performs better.

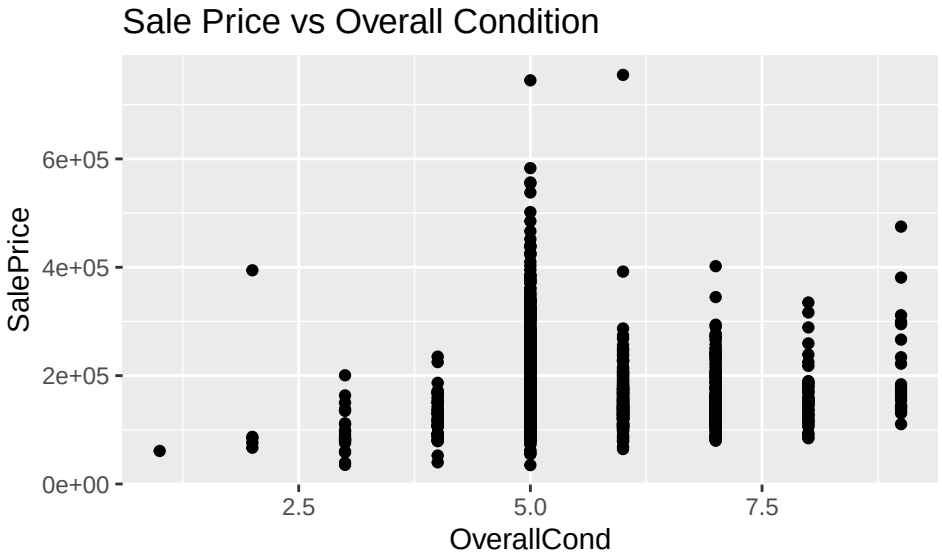
Exploratory

First we can analyze the correlation values between numerical variables. We can see the first few variable pairs with high correlation in the following table, which shows that many predictor variables have a strong linear relation with our response Sale Price. This informs our decision to fit a linear regression model, although we don't anticipate the linear regression model to perform best since the correlation values aren't extremely strong. This also suggests that a more complex model like trees or random forests may perform better at capturing this complex relationship.

```
##      row_name  col_name      corr
## 1 OverallQual SalePrice 0.7862117
## 2   GrLivArea SalePrice 0.7097950
## 3  GarageCars SalePrice 0.6361177
## 4  GarageArea SalePrice 0.6072095
## 5 TotalBsmtSF SalePrice 0.6029809
## 6   X1stFlrSF SalePrice 0.5959942
```

If we look at all of the predictor variables' correlation with Sale Price, we can observe some counter-intuitive trends. For example, the fact that Overall Condition is negatively correlated with Sale Price may seem strange since one would anticipate the price to increase as the overall condition of the house improves. However, if we look closely at the plot between Sale Price vs Overall Condition, we can see that this is a consequence of trying to fit a linear model on a discrete numeric variable. This is another indication that linear regression may not be the best choice, and we need a more complex model to capture these type of complex trends.

```
##        row_name  col_name        corr
## 1  KitchenAbvGr SalePrice -0.13743375
## 2 EnclosedPorch SalePrice -0.12032005
## 3   OverallCond SalePrice -0.09388637
## 4  BsmtHalfBath SalePrice -0.02484593
## 5        YrSold SalePrice -0.02378408
## 6    BsmtFinSF2 SalePrice -0.02225241
```



Next, if we look at the correlation between numeric predictor variables, we will notice that there are many pairs with high correlation. Serious collinearity concerns make fitting a full linear model inadvisable, so we use subset selection methods and LASSO regression methods to address this issue. Tree-based methods, particularly random forests, are also used since they perform better on datasets with correlated predictors. We also try manually removing variables that have over 0.5 correlation and see if that improves the performance of our models.

```
##       row_name     col_name      corr
## 1   GarageCars   GarageArea 0.8312173
## 2    YearBuilt  GarageYrBlt 0.8246693
## 3  TotalBsmtSF    X1stFlrSF 0.8211731
## 4    GrLivArea TotRmsAbvGrd 0.8209188
## 5    X2ndFlrSF    GrLivArea 0.6862310
## 6 BedroomAbvGr TotRmsAbvGrd 0.6647300
```

The categorical variables are also problematic for linear regression, especially since some variables like Neighborhood have 24 different levels. Since linear regression creates dummy variables for each level of a categorical variable, linear regression on this data set will soon create numerous amounts of dummy variables. This will make the linear regression models more expensive to fit and harder to interpret, which further certifies the need of model selection methods.

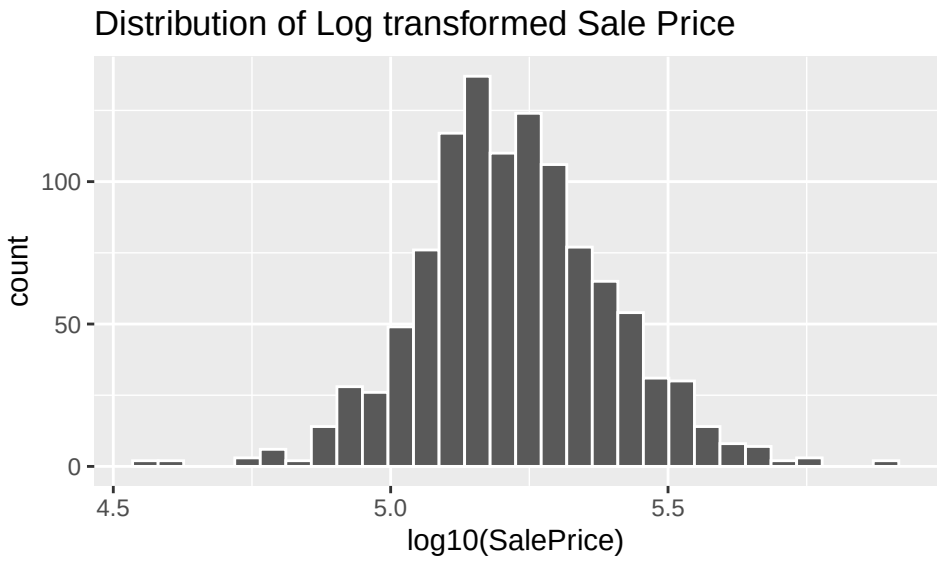
```
##    MSSubClass      MSZoning         Alley      LotShape   LandContour
##            14             5             3             4             4
##     LotConfig     LandSlope  Neighborhood    Condition1    Condition2
##             5             3            24             7             2
##      BldgType    HouseStyle     RoofStyle      RoofMatl   Exterior1st
##             5             8             5             3            11
##   Exterior2nd    MasVnrType     ExterQual     ExterCond    Foundation
##            11             5             4             4             5
##      BsmtQual      BsmtCond  BsmtExposure  BsmtFinType1  BsmtFinType2
##             5             5             5             7             7
##       Heating     HeatingQC    CentralAir    Electrical   KitchenQual
##             3             5             2             4             4
##    Functional   FireplaceQu    GarageType  GarageFinish    GarageQual
##             6             6             6             4             5
##    GarageCond    PavedDrive         Fence   MiscFeature       MoSold
##             4             3             5             3            12
##      SaleType SaleCondition
```

```
##              4              6
```

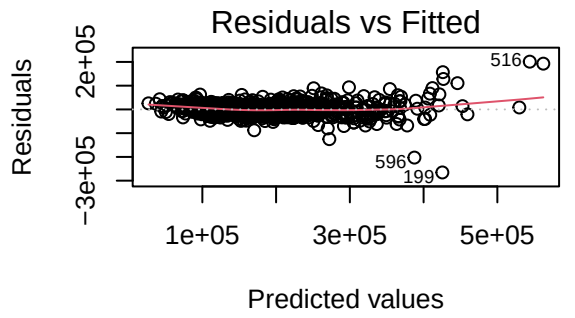
Finally, we can investigate the distribution of the response variable Sale Price. The distribution is clearly right-skewed with outliers that have extremely high sale price values, so linear regression methods may suffer. Hence we consider a log transformation on the response variable to de-emphasize outliers and obtain a more normally-distributed response variable. Moreover, the residual plots confirm that doing a log transformation produces a residual plot with more constant variance and less skewness. This suggests that taking the log might improve linear model performance. However, we should also beware that the residuals still don't follow a normal distribution even after log transformation, as shown by the Normal QQ plot.

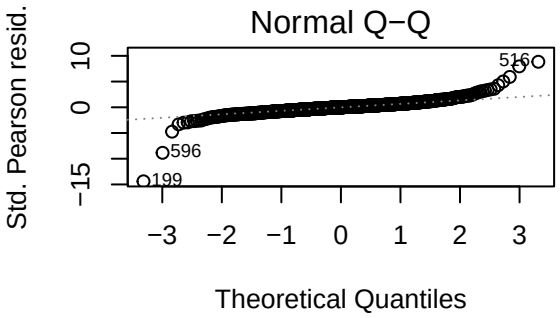
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

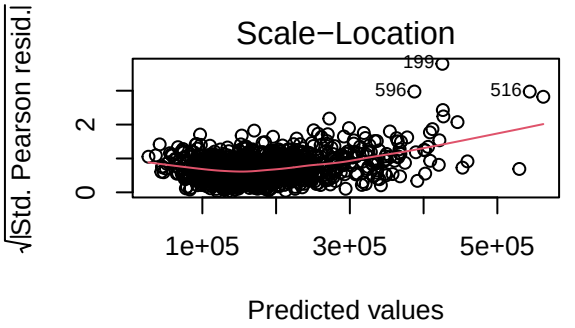

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

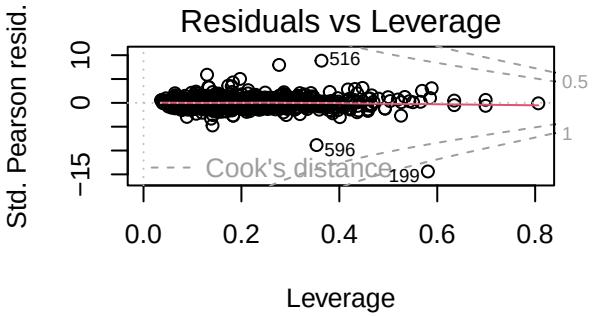



```
## Warning: not plotting observations with leverage one:
##   429, 957
```

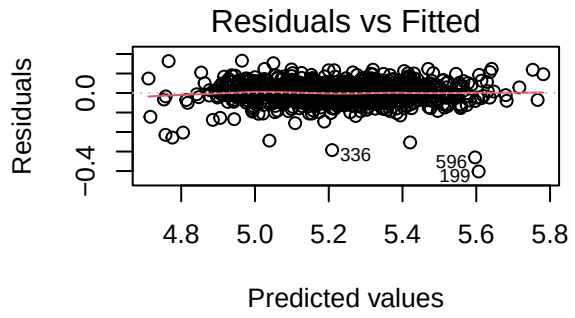


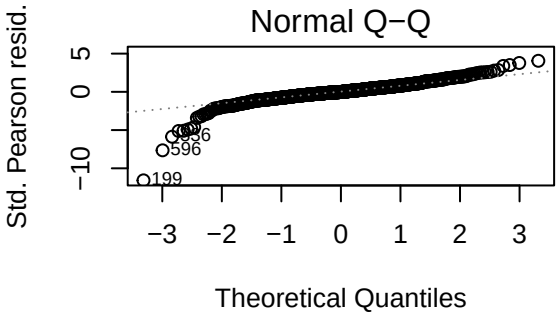


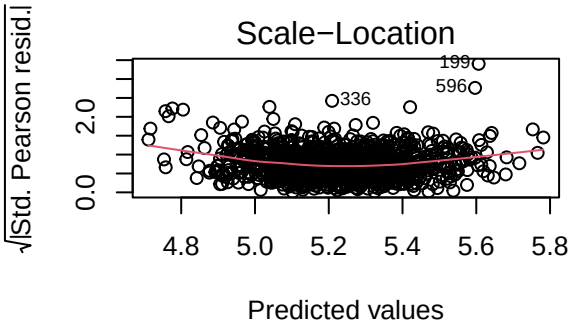


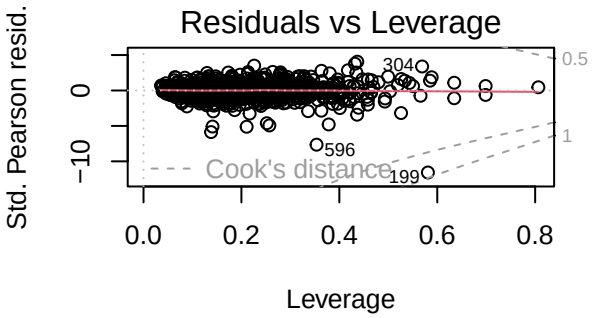



```
## Warning: not plotting observations with leverage one:
##   429, 957
```









LR with Subset selection

Reduce data

We compute the correlation matrix and use the findCorrelation function to drop variables that have over 0.5 pair-wise correlation. The way this function chooses which of the pair to drop is by computing its mean correlation value with all other variables, and dropping the one with a higher mean correlation value. We drop these variables in to create reduced train and test sets. The variables dropped are shown below.

```
##  [1] "GrLivArea"    "OverallQual"  "GarageCars"   "TotalBsmtSF"  "GarageArea"
##  [6] "FullBath"     "TotRmsAbvGrd" "YearBuilt"    "GarageYrBlt"  "X2ndFlrSF"
## [11] "BsmtFinSF1"
```

Forward Selection

We use forward selection on usual data, reduced data, log-transformed data, and reduced log-transformed data.

```
## Reordering variables and trying again:
```

```
## Reordering variables and trying again:
```

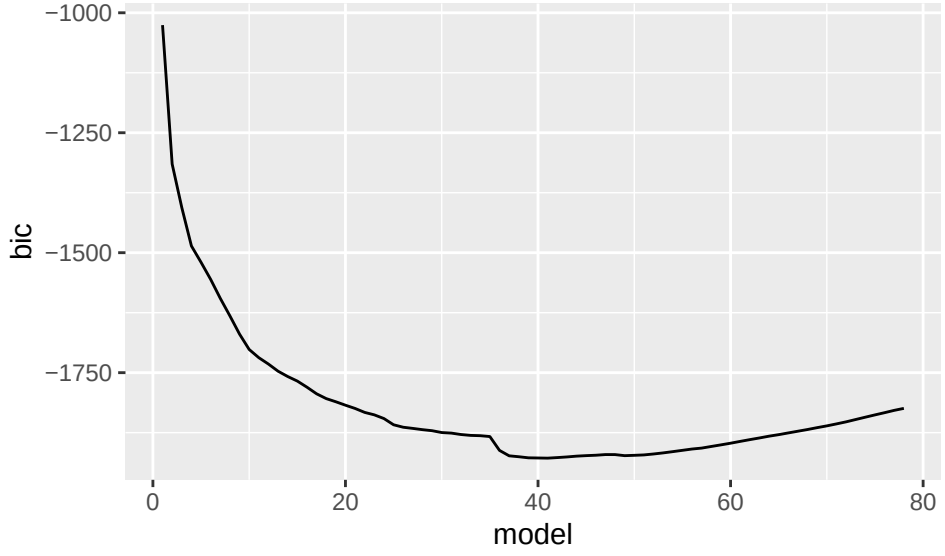
```
## Reordering variables and trying again:
```

```
## Reordering variables and trying again:
```

We choose the model with minimum BIC metric value, since all other metrics give the full model as the best model. For example, the forward selection on the usual data gives the following models as the best according to different metrics.

```
##   adjr2.max rss.min cp.min bic.min
## 1        78      78     78      41
```

Here is a plot of the BIC metric as model complexity increases. We can see that the minimum occurs at 41 variables.



Backward Selection

Similarly, we use backward selection on four types of data: whether or not it is reduced, and whether or not it is log-transformed.

```
## Reordering variables and trying again:
```

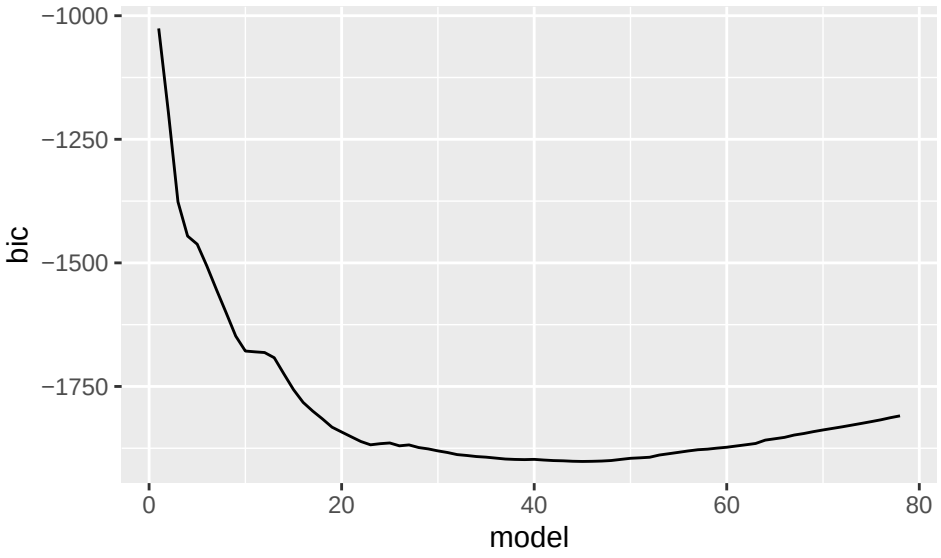
```
## Reordering variables and trying again:
```

```
## Reordering variables and trying again:
```


```
## Reordering variables and trying again:
```


We continue to use bic.min as the metric for model selection, since all other metrics give the full model.

```
##   b_adjr2.max b_rss.min b_cp.min b_bic.min
## 1          78        78       78        45
```



Predict

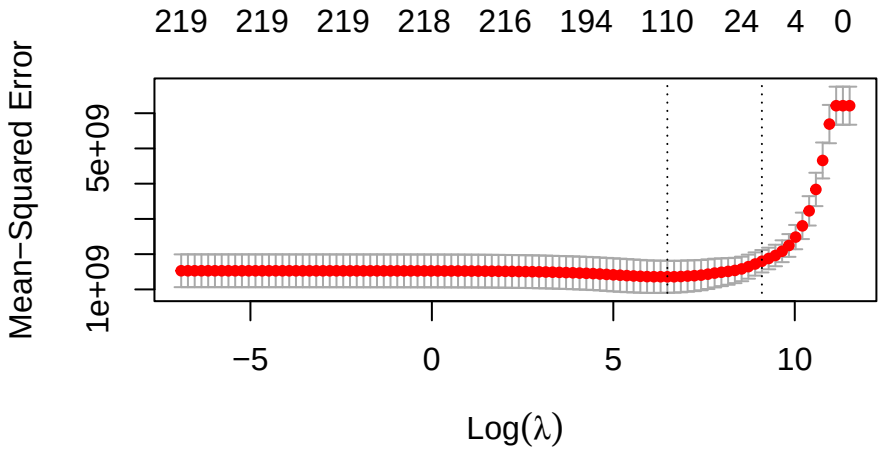
Next we make predictions and compute the test MSE values for each model.

We can see that both log transformation and reducing the dataset will reduce the test MSE values. This confirms our conjecture in the exploratory state that these methods will improve model performance. Overall, backward selection seems to outperform forward selection. Hence backward selection on log-transformed reduced data performs the best linear regression.

Algorithm	Test_MSE	Test_RMSE
Forward	1.602e+09	40031
Forward Reduced	1.929e+09	43920
Forward Log	1.398e+09	37390
Forward Log Reduced	1.123e+09	33508
Backward	1.404e+09	37467
Backward Reduced	1.179e+09	34342
Backward Log	1.334e+09	36527
Backward Log Reduced	1.059e+09	32536

Lasso regression

We apply lasso regression on both usual and logged data.



For our optimal λ values, we take the λ values that minimizes CV error:

We fit our LASSO and LASSO Log models with the optimal parameter:

We see that for LASSO, the log transformation also improves test MSE. We thus choose the logged model for LASSO.

Algorithm	Test_MSE	Test_RMSE
LASSO	755765449	27491
LASSO Log	657485003	25641

Tree-based methods

Training

Next we train our regression trees. We start with a single pruned decision tree. First we train the full tree:

Next, we pick the maximum value of CP that has X-relative error within one standard deviation of the mean. This value balances performance and complexity. This value of CP is given below:

Now we prune our tree with the optimal CP value to make our final tree:

Now that the pruned regression tree has been fit, we move on to bagged and random forests:

First we fit the bagged tree, letting `mtry` be equal to the number of predictors:

Now we fit the random forest. Here we set `mtry` equal to the default number for a random forest for regression ($p/3$).

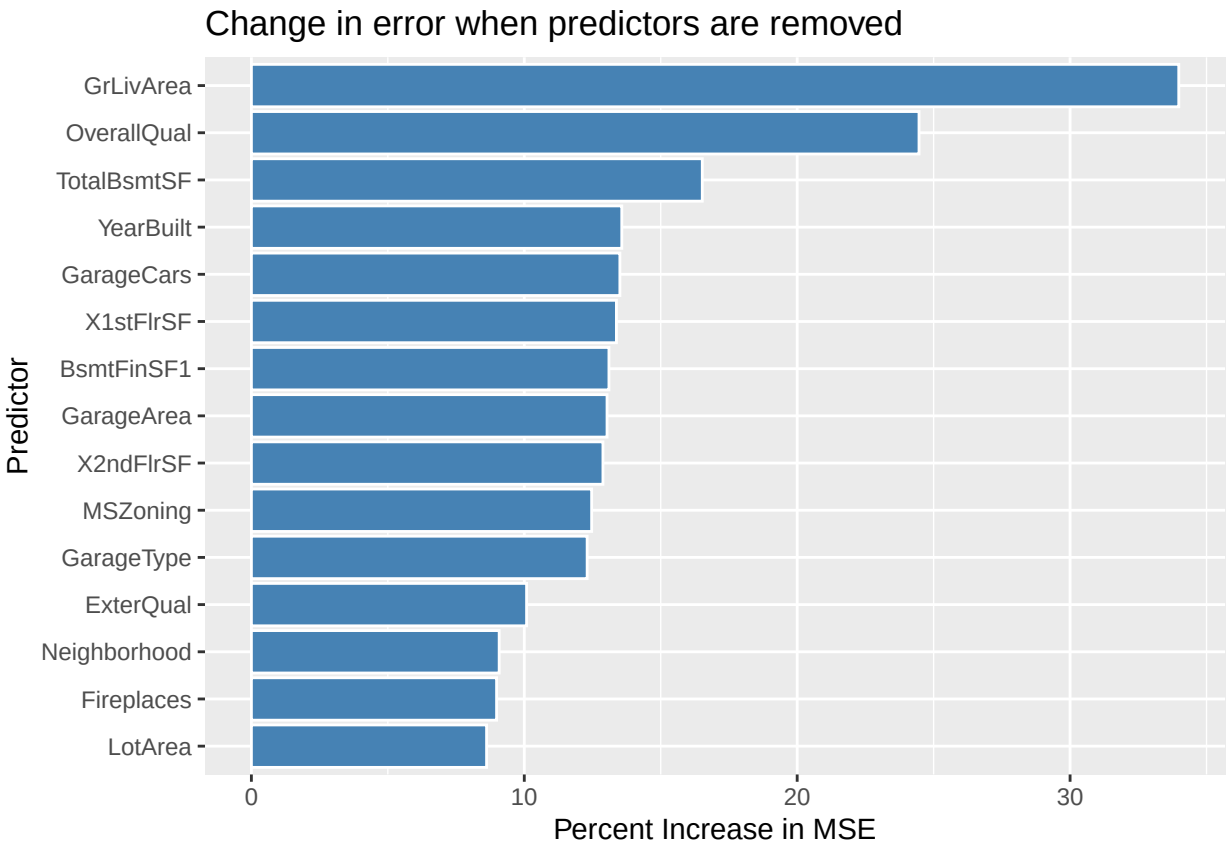
Prediction

Now we extract predictions on test data and create a table of MSEs for each model.

Algorithm	Test_MSE	Test_RMSE
Pruned Tree	1.436e+09	37893
Bagged Forest	655452033	25602
Random Forest	631122019	25122

Among trees, random forest performs the best and the pruned tree performs the worst. The fact that the pruned tree performs worse than the other two trees makes sense since pruned trees are notorious for overfitting. The random forest also performs about 1% better than the bagged tree. Whether this increase in

performance is significant is subject to debate, but we would expect a random forest to outperform a bagged tree on these data. This is because, returning to our EDA, many of the predictors in this dataset are highly correlated, which means that a bagged tree, which has access to all of the predictors at every step, might systematically choose nonoptimal trees due to its greedy selection algorithm. In addition to predictions, the random forest also gives us access to variable importance data which we investigate now.



Here the metric printed in the table is the percent increase in out of bag MSE when the values of a predictor are permuted. This metric provides a fairly robust estimate of the relative importance of predictors in our dataframe. Far and away the most important predictor is General Living Area. After that the next most important predictor is Overall Quality, a rating of the overall material and finish of the house on a scale one to ten. Year Built comes in 4th place, and many of the remaining important predictors have to do with the floor area and quality of various parts of the house. In general, it seems that the key factors in determining house price seem to be size, quality of material and finish, and the age of the home. Two other interesting important predictors are `Neighborhood` and `MSZoning`. Neighborhood is self explanatory, but `MSZoning` pertains to the general zoning classification of the sale (ie Commercial, Agricultural, Residential High Density, and so on). The importance of this variable seems to indicate that differently zoned transactions have different Sale Prices.

Compare all methods

```
##              Algorithm   Test_MSE Test_RMSE
## 1        Random Forest  631122019  25122.14
## 2            LASSO Log  657485003  25641.47
## 3 Backward Log Reduced 1058568888  32535.66
```

```
## [1] 0.02025352
```

From the table we see that the Random Forest model outperforms all other models with RMSE 25122.14.

Noted that LASSO Log model only has a 2% higher RMSE than the Random Forest. Given the large number of predictors, it's surprising how good LASSO Log performs against random forest. If the problem is inference instead of prediction, LASSO Log is preferred to perform variable selection.

Here we can easily print out the variables/levels selected by the lasso model with top 15 highest coefficients:

```
##                           coef
## NeighborhoodStoneBr 57202.366
## NeighborhoodNoRidge 56867.824
## NeighborhoodNridgHt 35925.303
## NeighborhoodVeenker 18756.103
## BsmtExposureGd      18171.593
## NeighborhoodCrawfor 17777.914
## NeighborhoodSomerst 16061.802
## LandContourHLS      13483.996
## SaleTypeNew         12982.869
## Exterior1stBrkFace  11813.640
## GarageCars          11697.137
## LotShapeIR2         11286.380
## OverallQual         10286.758
## LandContourLvl       9964.936
## RoofMatlother        8790.998
```