FIN 510: Final Project
EXECUTIVE SUMMARY

Your Team Name (be creative): Super gbm

Select whether this is an individual or group submission. **No more than 3 members per group.** Beyond the fact that all group members may submit the same answers, each submission must be separate work.

□ Individual Submission

■ Group Submission. Group member names: Hsuan-Min Huang (hmhuang2), Zih-Cian Fu (zcfu2), and Brian Wu (bhwu3)

Case Overview

In the United States, property taxes are collected by local governments and are used as an important source of funding for local construction and social welfare policies. The Cook County Assessor's Office (CCAO) is responsible for the valuation of properties in Chicago and 130 other municipalities, making it one of the most complex housing tax bases in the United States. Since CCAO has to estimate more than a 1.8million houses while taking into account various housing types (e.g., skyscrapers, bungalows, condominiums), CCAO's housing valuation has been highly inaccurate in the past. Therefore, CCAO has faced intense public pressure because of inaccurate and opaque information on housing valuations. But the good news is that Fritz Kaegi, who won the election in 2018, promised to reform the CCAO, improve the transparency of data, and create a data science team.

Objective

The objective of our project is to solve the housing tax problem of local governments by constructing a housing valuation model to increase the accuracy of housing price prediction.

Methodology

Gradient boosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is a weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function. The R package for the gradient boosting model is gbm.

Mean square error

In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a

more accurate estimate. In machine learning, specifically empirical risk minimization, MSE may refer to the empirical risk (the average loss on an observed data set), as an estimate of the true MSE.

Independent variable selection

Our choice of independent variables was based on the CCAO's Residential Automated Valuation Model. CCAO had a well-trained model and the importance of each feature in the historical data had been well tested. Therefore, we think it is reasonable to start with the variables presented in CCAO's valuation model and then make adjustments from there. There are 21 variables selected from the CCAO model, 3 variables added manually, 2 variables dropped due to too many missing values, and 4 variables not added that were in the CCAO model but not presented in our historical data. Therefore, we have a total of 24 independent variables used in our train model.

Data Manipulation

1. Dropping observations with NA: We dropped the observations that have NA values in four variables – 'geo_school_elem_district', 'geo_school_hs_district', 'char_type_resd', and 'char_air' because we thought they would have relative importance to our model, leaving them empty would seriously affect the performance of the model.
2. Filling values to NA: For NA values in variable 'econ_midincome', we fill in the median of the variable. And for the other variables, we fill in the modes of the variables as replacements for NA values.
3. Dropping variables with too many NA's: Variables with disproportionate NA values, 'char_tp_dsgn' and 'char_apts', are dropped since they might not have meaningful importance to the model due to too many missing values.
4. Changing categorical variables to factor variables: There are several integer variables in the historical data that should be either categorical or character variables, thus, we changed them into factor variables before building the model.
5. Adding new variables: There are three variables added, 'char_bldg_sf_poly_2', 'char_hd_sf_poly_2', and 'char_age_poly_2', as they are the squares of numeric variables 'char_bldg_sf, 'char_hd_sf, and 'char_age'. We also got this idea from the CCAO model.

Data splitting

We split the historical data into three randomly assigned data sets. First, we split the data into two parts: a testing set with 30% of the observations and a training set with 70% of the observations. And then, we further split the training set into a validation set with 25% of the observation and a training set with 75% of the observation.

Parameter tuning and assessment

In order to find the hyperparameters (shrinkage, tree depth, minimum number of terminal nodes, bagging fraction) for the gradient boosting model, we perform a grid search in R which iterates over every combination of hyperparameter values and allows us to assess which combination tends to perform well. In this case, we are going to search through 81 combinations with varying learning rates and tree depth, and apply 5000 trees for each of them, training the model through validation set and training set. The article we cited took 30 minutes

to tune the parameters. However, it took us three to four hours (which could vary from machines) to finish the grid search, we suspect that this could be caused by the fact that the size of our data was way larger than the author's data. After the search was finalized, we picked the set of parameters that resulted in minimum MSE on the validation set and used them to build our gradient boosting model for our testing set and the final predictions of the sale price for the properties.

Model evaluation

First, we visualize variable importance. Variable importance is measured by how well a variable could influence MSE. The variables with the largest average decrease in MSE are considered the most important. As expected, variables 'meta_nbhd', 'char_bldg_sf', 'school_elem_district', 'char_age', and 'char_fbath' all have relatively high importance in the model, whereas 'school_hs_district' and 'meta_town_code' were having a surprisingly low influence to our model.

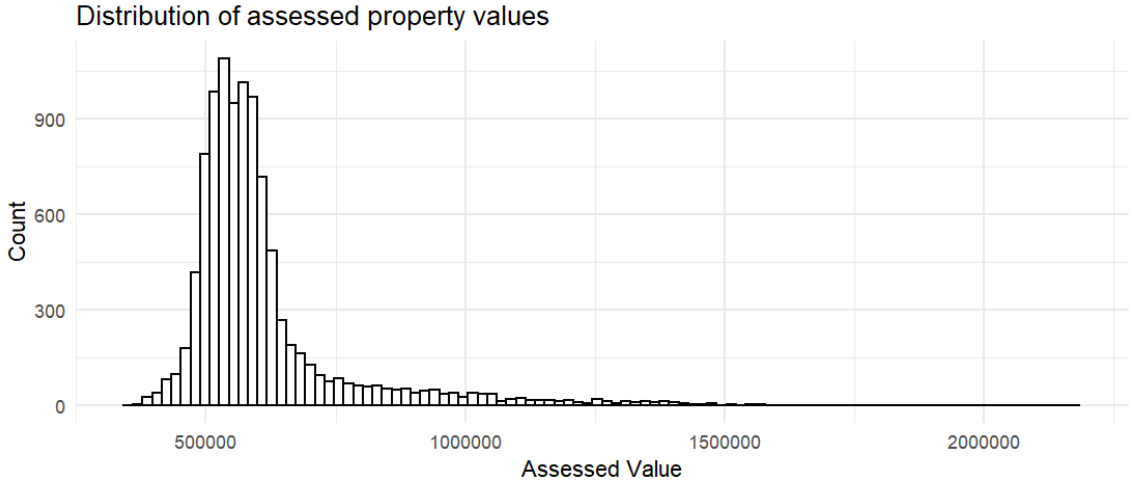
Secondly, we perform a Mean Square Error test on the testing set, through which we got a result of 19,895,540,786.

Conclusion

The results of our summary statistics of 10,000 predicted home prices are as follows:

```
> summary(assessed_value$assessed_value)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 359393  522924  568937  614093  622324 2185800
```

From the above graph, the minimum value is 35393, the maximum value is 2185800, the mean is 614093, the median is 568937, the 25th percentile is 522924, and the 75th percentile is 622324. The following distribution of assessed property values is a histogram that divides the forecast results into 100 equal parts. According to the histogram below, the distribution of assessed property values is right-skewed because the mode (approximately equal to 518264) < median (568937) < mean (614093). In other words, there are a few properties that have a much higher forecast price than the average house forecast price of 614093.



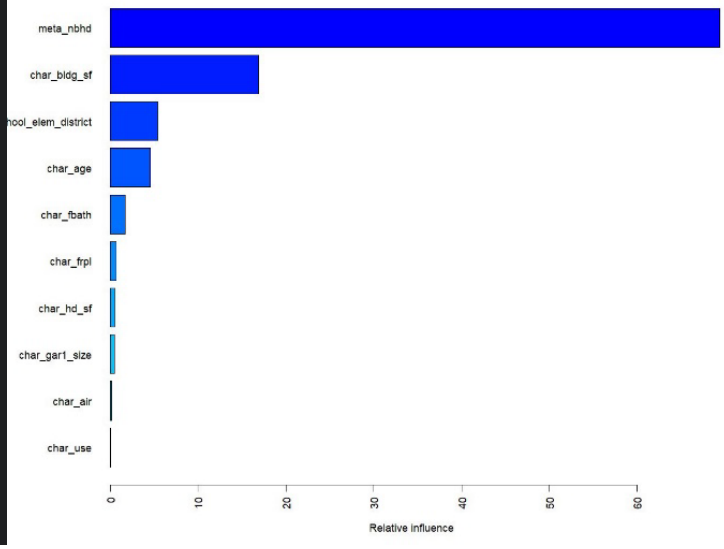
Appendix

The following are the top 10 parameters that we obtained using the gbm algorithm and the training dataset. The parameters of the first row are the parameters that we put into the gbm.fit.final function later.

```
> hyper_grid %>%
+   dplyr::arrange(min_MSE) %>%
+   head(10)
   shrinkage interaction.depth n.minobsinnode bag.fraction optimal_trees min_RMSE     min_MSE
1       0.01                 5             15         1.00           437        0 20116381767
2       0.01                 5              5         1.00           831        0 20552719045
3       0.01                 5             15         0.80           342        0 20678805333
4       0.01                 5             10         1.00           326        0 20927251134
5       0.10                 5             15         1.00            37        0 20931785011
6       0.10                 5             15         0.80            32        0 20958773263
7       0.01                 5             15         0.65           308        0 21138809459
8       0.10                 5              5         1.00            74        0 21163154338
9       0.10                 5             10         1.00            97        0 21174593191
10      0.01                 5             10         0.80           344        0 21182447990
```

If we use the summary function in our property price prediction model, the following results will be obtained. The following results show that meta_nbhd has the largest relative influence in this model, followed by char_bldf_sf. The right figure shows the relative influence in descending order.

```
> summary(
+   gbm.fit.final,
+   cBars = 10,
+   method = relative.influence,
+   las = 2
+ )
                                       var      rel.inf
meta_nbhd                         meta_nbhd 69.405002919
char_bldg_sf                   char_bldg_sf 16.895873929
geo_school_elem_district geo_school_elem_district  5.424642468
char_age                           char_age  4.556122528
char_fbath                       char_fbath  1.706506151
char_frpl                         char_frpl  0.714257347
char_hd_sf                       char_hd_sf  0.528002201
char_gar1_size                 char_gar1_size  0.461826950
char_air                           char_air  0.108093422
char_use                           char_use  0.099623955
econ_tax_rate                 econ_tax_rate  0.088765639
char_rooms                       char_rooms  0.007571325
econ_midincome                 econ_midincome  0.003711168
geo_school_hs_district geo_school_hs_district  0.000000000
meta_town_code                 meta_town_code  0.000000000
char_beds                         char_beds  0.000000000
char_type_resd                 char_type_resd  0.000000000
char_ext_wall                   char_ext_wall  0.000000000
char_bsmt                         char_bsmt  0.000000000
char_hbath                       char_hbath  0.000000000
char_gar1_att                 char_gar1_att  0.000000000
poly_char_hd_sf             poly_char_hd_sf  0.000000000
poly_char_age                 poly_char_age  0.000000000
poly_char_bldg_sf         poly_char_bldg_sf  0.000000000
```



Finally, after calculation, the MSE of our property valuation model is 19,895,540,786.

```
> MSE <- mean((pred-feature_df_test$sale_price)^2)
> MSE
[1] 19895540786
```