**A picture containing text

Description automatically generated**

Final Project

Predicting How Long It Will Take Properties to Sell in Chicago?

BUSN 41204-01 - Machine Learning – Winter 2022 Prof. Mladen Kolar

| **Group** |
| --- |
| Doug MacIntosh |
| Anna Jacobs |
| Anthony Ivy |

**Honor Code Acknowledgement**

“We pledge our honor that we have not violated the Chicago Booth Honor Code during the preparation of this assignment.”

1. **Executive Summary**

Understanding the time it takes to sell a property in Chicago creates a path to lucrative opportunities in the real estate market. Failing to realize how long it will take you to recoup a return on your investment when flipping properties is how people lose money in real estate. Purchasing properties, renovating them, and selling them to investors can make the seller substantial money. However, if an investor buys a property and can sell it after it is fixed for a long period, the person is at risk of losing a substantial amount of profit or can take a loss. Our objective is to create a predictive measure that can be used to understand how long it will take an investor to sell their investment property. This approach effectively removes the risk of in the business of property flipping.

1. **Background**

The property flipping business can be very lucrative, but many investors complain that they lost their money in real estate.  The universal goal of investing is to purchase a property that appears to be profitable when accounting for the purchase price and the capital expenditures necessary to bring the property to market.  The amount of time it takes to sell the property after the investment is usually a guessing game for the investor.  Properties can take anywhere from 14 to 600 days to sell.  The most common reason for an investor to lose their money or creditworthiness is to buy a property that they estimate would take less than 30 days to sell, and the property would take more than 90 days to sell.  Thanks to improvements in machine learning technology, property flippers can take the guesswork out of knowing if their property will sell quickly and make them a profit.

1. **Dataset**

Realtor MLS Data

In this paper, we used the dataset publicly available from the MLS Realtor Database ([1]). The data contains 3,000 properties that have sold from January 1st, 2021, to December 31st, 2021.  The list of properties are buildings that are 2-4 unit buildings across the Chicago Metro Area.  The Data set is robust with 131 feature variables and one label column for the property addresses.  The MLS data has all the information about the properties that help differentiate each property.  Items total units, property size, and types of fixtures.

United States Census Bureau Data

The data set is also mixed with United State Census Bureau data. We included this data to provide more color to the data set and include information not readily available in the MLS. Much of this information is secondary demographic information, and the goal is to provide additional information that the property-focused MLS data does not provide. Most of the Census Bureau data is predominantly demographic-focused and describes the environment or actions of the people that live in the different areas of the properties. The Census information was coupled with the MLS data by attaching the additional columns to the area column of the MLS data. Below is an example of the column descriptions included in the Census data set.Table

Description automatically generated

1. **Methodology**

Overall Strategy

In this paper, we attempt to predict the time it will take to sell a property using many machine learning algorithms.  We created a factor column that categorizes the data as follows:

* Property took 0-14 days to sell
* Property took 14-30 days to sell
* Property took 30-60 days to sell
* Property took 60-100 days to sell
* Property took 100+ days to sell

By segmenting out the columns, we created a prime target variable that the machine learning algorithms could train with to predict whether a home introduced to the models would give an accurate prediction of how long it will take for the property to sell. We went about the research by taking the following steps:

1. Preprocessing the data
2. Performing a Boruta Algorithm to indicate the important factors of the dataset
3. Performing PCA and Cluster analysis on the data set to understand how the factors play out and segment the data set into unique markets
4. Running machine learning algorithm on the data set (Random Forest, XGBoost, Decision Trees, and Vector machines)
5. Exploring the models to select the smallest Root Mean Squared Error (RMSE), highest Area Under Curve (AUC), and highest accuracy
6. Testing the chosen machine learning methods on the out of sample data
7. **Data Preprocessing**

To perform the machine learning research, we first had to pre-process the data. The process was extensive given the number of factors and missing data points. Much of the imported data showed up as characters, so we first numerated the integers and dollar-related columns. Five columns were data-related, notably, the closing dates of the sales and the dates the properties were added to the MLS for sale. We changed all of the date columns into standard date columns.

Next, we combed through all of the factors and gave them their property type classification so that the algorithms were getting all the information needed from the data set. After we classified the data, we noticed that many columns were characterized as integers but contained less than 22 levels. We transitioned those items into factors as well. Finally, we input all of the missing numerical values with the median so that we did not have any missing items that would deter the algorithms. Finally, we ran an aggregate plot to show if there were any missing values. We followed this by applying a skim function to ensure that all features were correctly classified, and the data set were in order. The aggregate plot is below:Chart

Description automatically generated

1. **Boruta**

The first algorithm we ran on the data set was a Boruta. The goal of running this was to filter the data set down to the most impactful variables. The Boruta Algorithm serves as the inference detection so that we can see clearly which columns have the most impact on property sell time in the marketplace. The findings were very robust, and out of the 131 feature variables we were able to reduce the data set to the most important factors. In total, about 23 factor variables were significant of medium importance or high importance. These 23 factors are factors we chose to continue forward into the machine learning algorithms. Some of the factors that had the most impact on the data were as follows:

* The property being on the north side of the city or the south sides specifically.
* The among of high school and associate graduate degree holders
* The amount of white demographic residents living in the area
* The employment rate of the city
* **Chart

  Description automatically generated**The number of residents that work from home in the area

1. **Principal Component Analysis**

The next step of our analysis was using principal component analysis, MBC lust measurements, and gap stat to figure out how my clusters to use - our original analysis giving mixed results. We drew a Scree Plot which had a very defined elbow at the 3 principal components with a marginal increase up to 10 components. Chart, line chart

Description automatically generatedWe plotted the results defined by our Target\_Sell\_Time variable. We learned very little information from that exercise.

Chart, line chart

Description automatically generated

We proceeded to complete an NBC lust measurement for our clusters. This suggested that 2 clusters would be sufficient.

Chart, line chart

Description automatically generated

Finally, we ran a gap stat, which suggested 10 clusters as the most appropriate for the Chicago real estate market. For gap state, as K goes up, error goes down.

After thinking about our options, we decided to proceed using the 10 clusters suggested by gap stat because 1) this unsupervised learning algorithm is supposed to have high accuracy and 2) In practice this made sense as the Chicago real estate market has more than 2-3 distinct groups.

Chart

Description automatically generated

We ran an auto plot using the 10 recommended clusters with combined data from the original data set to arrive at the PCA Cluster map you see above. This PCA cluster map not only shows us the direction that the loading is pointing, but also the accompanying clusters that are affected by these loadings. Looking more closely at the loadings, we see there is significant overlap with the high and medium importance variables that we received from running the Boruta analysis.

Map

Description automatically generatedOur final step in understanding how the target\_time\_to\_sell variable relates to the clusters, we mapped the clusters onto a Chicago map using the longitude and latitude of the properties that are in each cluster.  The map is as follows:

1. **Estimation of Predictive Model**

Now that we have a better idea of how our dataset concentrates in clusters around the Chicagoland area, we come back to our original goal of attempting to predict property sell times using machine learning models. Our initial hypothesis for the best models to use were tree-based models, such as Random Forest, XGBoost, and Decision Trees, and Support Vector Machines. These models are most appropriate since our target variable has distinct categories, making this more of a classification problem rather than a regression problem.

1. **Initial Model Development**

As discussed previously, we used Boruta variable selection to narrow down our variables to the most impactful towards our Target\_Sell\_Time classification. After that, we broke out the dataset to an 80/20 split for the model’s training and testing set. With our data ready, we began developing our models. For tuning the models, we created “for” loops and the tune function to iterate through each hyperparameter for the tree and SVM models. Those tuning functions output each of the best models as decided by RMSE. Upon reviewing the predicted output confusion matrices from each, we discovered a problem. Our models were inaccurate and heavily skewed towards “Sold in Less than 14 Days”. See below for an example. 

Initial SVM Confusion Matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Predicted/actual | False | <14 Days | 14-30 Days | 30-60 Days | 60-100 Days | >100 Days |
| False | 0 | 0 | 0 | 0 | 0 | 0 |
| Greater than 100 Days | 0 | 0 | 0 | 0 | 0 | 0 |
| Sold between 14 and 30 Days | 0 | 0 | 0 | 0 | 0 | 0 |
| Sold between 30 and 60 Days | 0 | 0 | 0 | 0 | 0 | 0 |
| Sold between 60 and 100 Days | 0 | 0 | 0 | 0 | 0 | 0 |
| Sold in Less than 14 Days | 1 | 382 | 154 | 123 | 100 | 137 |

After further analysis of the models and the data set, we concluded that our models and code had no errors that would cause this effect, but rather our dataset was heavily imbalanced. See below for the output of the original dataset.

Target\_Sell\_Time Counts from the original dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| False | <14 Days | 14-30 Days | 30-60 Days | 60-100 Days | >100 Days |
| 4 | 1275 | 516 | 413 | 335 | 457 |

1. **Initial Model Development**

Once this was discovered, we began troubleshooting ways we could correct it. At this point, we opted to try limiting the number of classes in Target\_Sell\_Time to help. Rather than use all six categories, we decided to split the variable into two segments: greater than 30 days and less than 30 days. This effectively makes our Target\_Sell\_Time variable binary and opens up the use of other model types. Also, it split the data into a much more even set (1112 samples greater than 30 days and 1288 less than 30 days). Following this change, we brought our data into the H2O Flow module to experiment with different models. We used their AutoML function to run through a variety of models and decided on using the following four models for this analysis: Deep Learning, Gradient Boosting Machine (GBM), Distributed Random Forest (DRF), and Multiple Linear Regression (MLR). The AutoML tuned each of the models to output the lowest AUCs. From this set of models, we chose the best highest AUCs of each model type for the basis of evaluation. See the Appendix for the output of each model and the ROC plot and predicted model values for our DRF model.

DRF ROC Plot DRF Predicted Values

Chart, line chart

Description automatically generated Table

Description automatically generated

1. **Test Results**

From the results, we see the tuned Random Forest performed the best via AUC and accuracy values (0.616750 and 0.5983). Though, this was only marginally better than the boosted model. This result could make sense from a conceptional standpoint with our imbalanced data set. Tree-based models operate by repeatedly resampling random samples of features within the dataset. This feature could potentially offset some of the error rates caused by the imbalance. It is slightly surprising that random forest outperformed the boosted model when running the prediction function. It may be an anomaly since the boosted model outperformed it during the model selection process. Regardless of that, both models significantly outperformed the Deep Learning and MLR models, as expected. Across all models, we still saw a significant prediction imbalance within our confusion matrices.

1. **Consideration of Resampling Methods**

After reviewing the AUC and accuracies for the models, we decided to try upsampling and downsampling our dataset to see if that improved our accuracies. For the model, we chose our tuned boosted model from the previous model selection. See below for the resampled dataset values for Target\_Sell\_Time.

Resampled Dataset values

|  |  |  |  |
| --- | --- | --- | --- |
| Upsampled | | Downsampled | |
| Greater than 30 days | Less than 30 days | Greater than 30 days | Less than 30 days |
| 1112 | 1112 | 1288 | 1288 |

After running our two models, we noticed an interesting change. The upsampled AUC and accuracy were much better than the downsampled. Also, our confusion matrices appeared to not improve from the previous models. After reviewing both model outputs, we concluded that not only is our dataset still significantly imbalanced even with our resampling effort, but also our dataset is extremely limited in scope.

13) **Conclusion**

In this paper, we attempted to create a model

time series

communities that lean away from selling sooner

communities that

1. **Conclusion**

In this paper, we tried to develop a model to predict intraday and next day’s price movement of Dow Jones Industrial Average (DJIA). We employed two strategies to accomplish this predic- tion, one of which is the estimation by using the sentiment index, and the other is by fitting with the document term matrix. While performed well in validation sample, fitting on the sentiment index was not a good strategy given the number of observation. However, if a model is fitted with document term matrix by Random Forest algorithm, the performance of the model was stable during the period of the test sample. As the stock market is complex and highly volatile, an algorithm which focuses on minimizing the variance would be more suitable to formulate the predictive model. If we construct a portfolio strategy which uses the prediction for the investment decision making, we could see the performance of such strategy is better than the DJIA index. This implies that if we choose an algorithm to model a prediction, we would be able to extract an useful information from a news feed.

Table 1: Evaluation Metrics on Validation Sample for Intraday Prediction

Appendix

Boosted Model Output

Chart, line chart

Description automatically generated

Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated

Neural Network Model Output

Chart, line chart

Description automatically generated

Table

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Random Forest Model Output

Chart, line chart

Description automatically generated

Table

Description automatically generated

Graphical user interface, table

Description automatically generated with medium confidence

Table

Description automatically generated with medium confidence

Table

Description automatically generated

Table

Description automatically generated

Multiple Linear Regression Model Output

Chart, line chart

Description automatically generated

Table

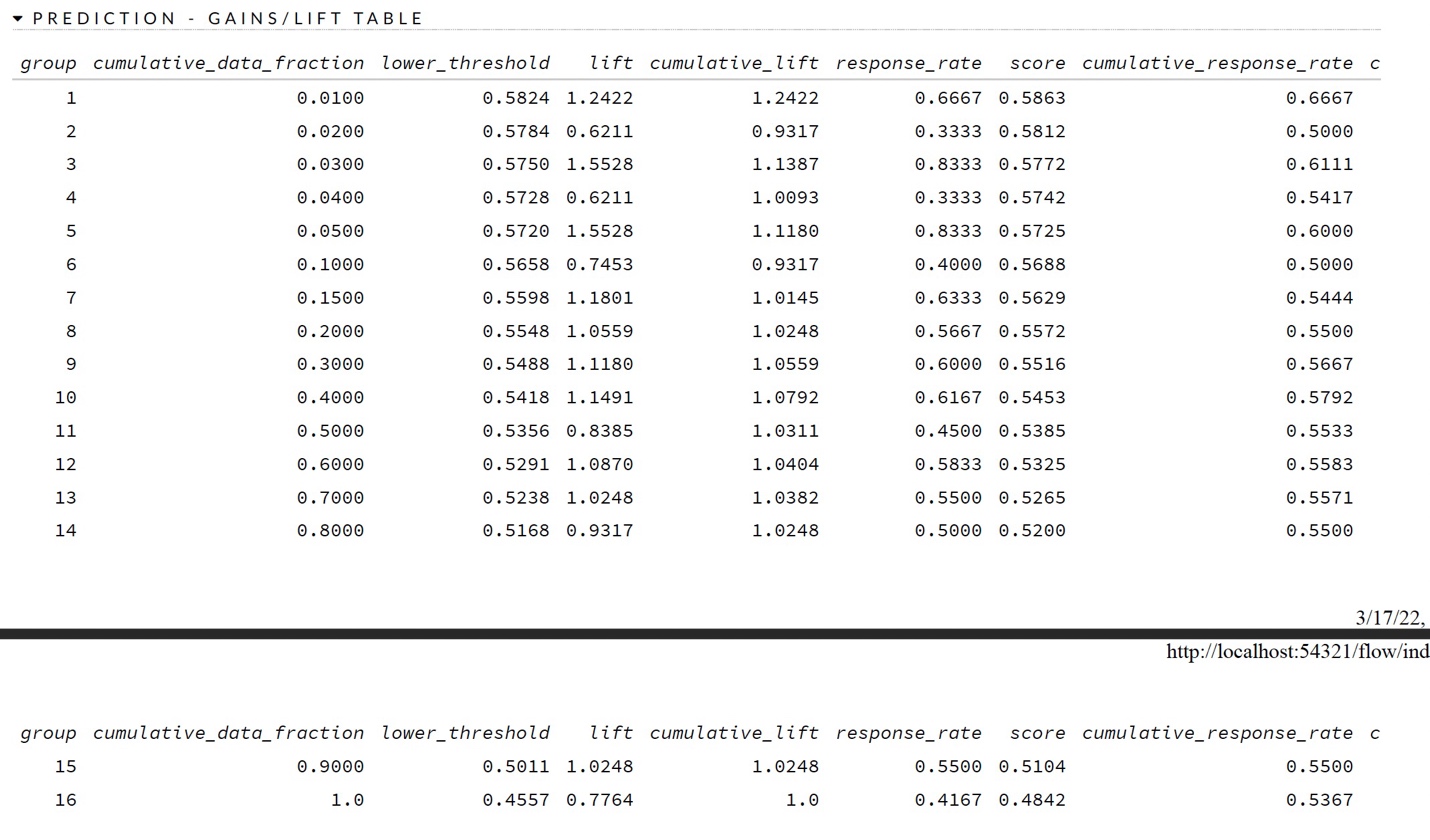
Description automatically generated

Graphical user interface, table

Description automatically generated

Table

Description automatically generated



Boosted Model Output using Upsampled Dataset

Chart, line chart

Description automatically generatedTable

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenceTable

Description automatically generatedTable

Description automatically generated

Boosted Model Output using Downsampled Dataset

