Stat 444/844 Final Project

Hieu Nguyen 20532698
Alice Li 20586843
April 13, 2019

Contents

Executive Summary

Based on the our findings, there are some interesting factors that play an important role in determining the pricing of a property. Just to name some of these factors that really stand out in our work. These includes the number of days from the sales date to today's date, the longitude of the property, the sale year, the relative spherical distance between properties, the gross building area in square feet, how old is the property in years since the earliest time the main portion of the building was built, the number of bathrooms, the number of fireplaces, etc. What even more interesting is that these important features have relatively high correlation with the price it self, which justifies their roles in our predictive process. In this project, we design 4 different models, all being trained and validated using 5-fold cross validation with an attempt to best predict the housing price. The first model is smoothing, which has the error (RMLSE) of 0.19856. The second model is random forest, which produces the error (RMLSE) of 0.180. Third, our boosting model has an error of 0.16746. Lastly, we took the average of the predictions from random forest and boosting and get an error of 0.17162. As far as we concern, the boosting algorithm clearly outperforms other models by a good margin. Overall, despite the fact that boosting takes a long time to train, we believe that it is the best candidate given the prediction errors and sensitivity to over fitting.

1 Introduction

The financial sector is in the midst of a transformation. The Canadian industry, long dominated by the "Big Five" (RBC, CIBC, BMO, TD, and Scotia), now has to compete with international banks and novel technologies such as robo-advisors. Coupled with the rumours of an impending North American recession, it is crucial that financial insistutions identify ways to identify and retain a loyal customer base. Targeted phone-based campaigns have long been useful tools to this end.

In this project, we aim to discover the attributes of a successful telephone-based marketing campaign promoting financial services. In particular, we are interested in identifying the characteristics of clients who have subscribed to a term deposit with a bank after being contacted by a customer engagement representative. We will be utilizing the outcome of a campain led by a Portuguese bank between 2008 and 2010.

2 Data

We analyzed the "Bank Marketing" dataset that was originally collected by Moro et al. for their 2011 paper "Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology". The researchers collected the data from a Portuguese bank that routinely performed telephone-based marketing campaigns between 2008 and 2010. They donated a reduced data set comprised of 11,162 observations and 17 variables.

We are interested in developping a model to determine whether or not prospected clients will make a deposit with the bank. Thus, we will use the binary "Deposit" variable as our response, meaning our classification model will have two classes. A priori, both classes are roughly proportioned equally: 52% of observations are of type "No" will the other 48% are of type "Yes". The other 16 explanatory variables are composed of .

The data was seemingly published entirely cleaned. There are no missing observations, although some data were inputted as "Unknown". None of the variables exhibited bivariate correlation superior 0.2 aside from "pdays" (the number of days that passed between the latest two campaigns in which the client was contact) and "previous" (the number of campaigns where the client was contacted before the current campaing). While the correlation is 0.51, this is to be expected: over two third of clients had only been contacted once, meaning they had pdays=-1 (code for observations that were not previously contacted) and previous=0.

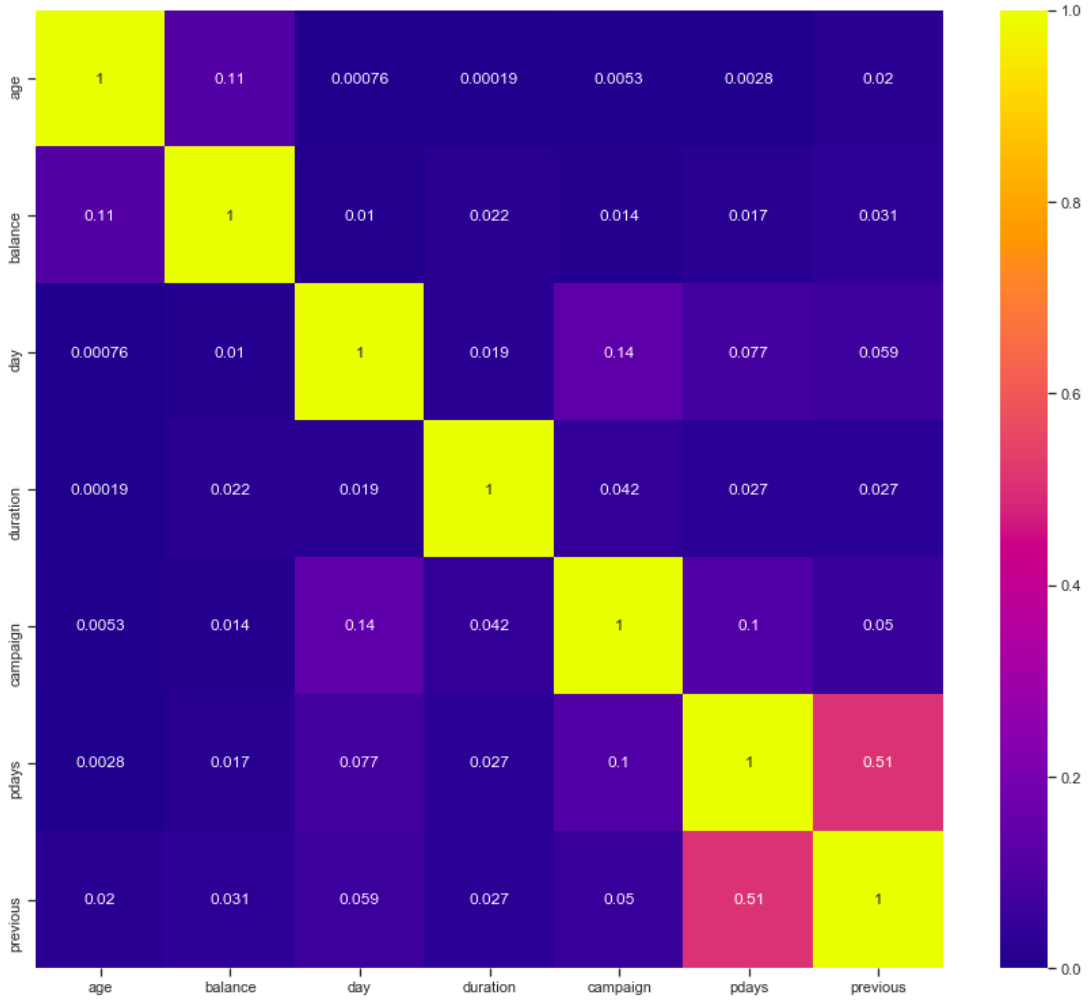

Figure 1: Correlation between numerical variables

3 Preprocessing

3.1 Missing data

Within the dataset, it's found that features *AYB*, *YR_RMDL*, *STORIES*, *KITCHENS* and *QUADRANT* contain missing (NaN) values. Most notably, 16351 out of the 39520 entries are missing within the *YR_RMDL* column. We replace missing values in *AYB* with the average of the rest of values, and replace missing values in *YR_RMDL* with the larger year value among *AYB* and *EYB*. This decision is based on the observation that 99% of time the value under *YR_RMDL* is larger than those in *AYB* and *EYB* given a house Id.The integer/float numbers contained within the strings under the column *STYLE* are used to replace missing values in *STORIES*. Comparison between columns *STYLE* and *STORIES* is further discussed in the following Imputation subsection. The missing values in *KITCHENS* are replaced by

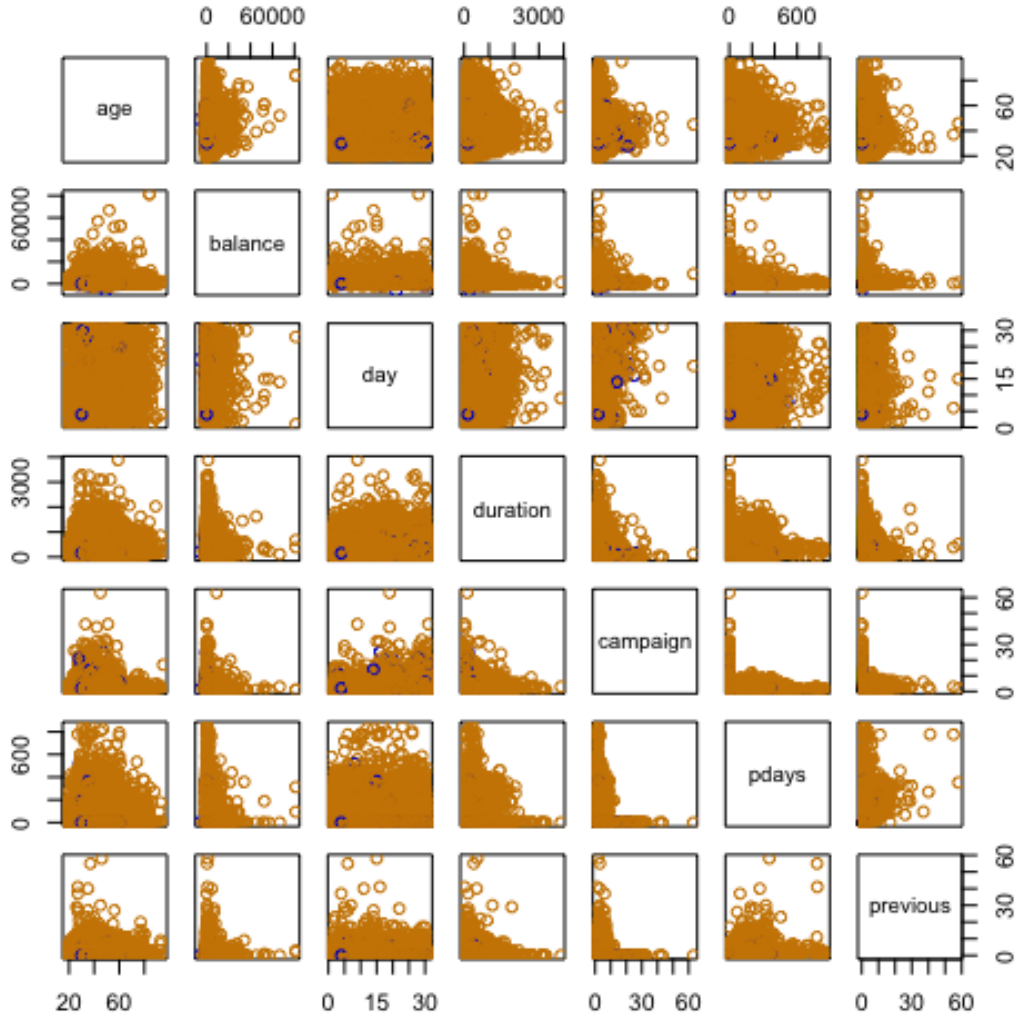

Figure 2: Plot of the data. Orange represents "No Deposit", blue represents "Yes Deposit". There is severe overlap between both classes for every variables.

its mode. Lastly, we replaced the missing values with the most frequently appeared *QUADRANT* value given the *WARD* value.

3.2 Imputation for Numerical Columns

Looking at the summary (mean, median, min, max) of the numerical features, we observe that under the *YR_RMDL* column, Id 21997 has a value of 20, and Id 27034 has 2019, which are incorrect entries given that this column contains integer values representing years before (and including) 2018. We replace the former with 1929, the

	Age	Balance	Day	Duration	Campaign	PDays	Previous
Count	11,162	11,162	11,162	11,162	11,162	11,162	11,162
Mean	41.23	1528.53	15.65	371.99	2.50	51.33	0.832557
St. Dev	11.91	3225.41	8.42	347.12	2.72	108.75	2.29
Min	18	-6847	1	2	1	-1	0
Quartile 1	32	122	8	138	1	-1	0
Quartile 2	39	550	15	255	2	-1	0
Quartile 3	49	1708	22	496	3	20	1
Max	95	81,204	31	3881	63	854	58

Table 1: Descriptive Statistics for Selected variables

median value of the column, and replace the latter with 2018 to ensure consistency with the data set description.

We also observe through the feature summary that Id 2322, 7435, 8056 correspond to unreasonable *STORIES* values, equalling 826, 250 and 275. We replace these values with 4, 2.5 and 3 respectively according to the information contained under the *STYLE* column. This finding further prompted us to investigate in the relationship between the *STYLE* and *STORIES* columns. We find that 39282 out of 39520 entries under *STYLE* contain numeric values representing the number of stories in the housing property. The difference in story numbers within these 2 columns are within ± 2 for all but 23 entries. We replace the values under *STORIES* with the numeric value within *STYLE*. We also 0 values under *STORIES* with 2, the corresponding value in *STYLE*. We also replace 0 values under *LANDAREA* with the average of the column, as it is not intuitive to have a property occupying a land of zero size.

3.3 Imputation for Categorical Columns

Firstly, we replace the 0 values under the *AC* column with 'N'. For each categorical column (variable), we count the frequency each category occurs within the data set. It is found that, under column *HEAT*, more than 95% of the data points can be grouped under categories 'Forced Air', 'Hot Water Rad' and 'Ht Pump'.To reduce computing time, we re-categorize data points that do not fall into the above categories into a separate 'Other' category. The same re-categorization procedure is repeated for the *STRUCT* column for points not in categories 'Row Inside', 'Single', 'Semi-Detached' and 'Row End'; for the *EXTWALL* column for points not in 'Common Brick', 'Brick/Siding', 'Vinyl Siding' and 'Wood Siding'; for the *ROOF* column for points not in 'Comp Shingle', 'Built Up', 'Metal- Sms' and 'Slate'; for the *INTWALL* for points not in 'Hardwood', 'Hardwood/Carp', 'Wood Floor' and 'Carpet'.

We re-categorize the *GRADE* column as following. First we regroup 'Exceptional-A', 'Exceptional-B', 'Exceptional-C' and 'Exceptional-D' under a category 'Exceptional',

merge categories 'Good Quality' and 'Above Average' into 'Above Average', and merge 'Fair Quality', 'Average' and 'Poor Quality' into 'Average'. Secondly, assign order Average → Above Average → Very Good → Excellent → Superior → Exceptional to the the *GRADE* category according to the box plot below. Similarly, under the *CNDTN* column, we recategorize 'Fair' and *CNDTN* to 'Average', and assign the order Poor → Average → Good → Very Good → Excellent. The ordinal property of these two columns is found very useful in improving the prediction accuracy using Random Forest and Boosting.

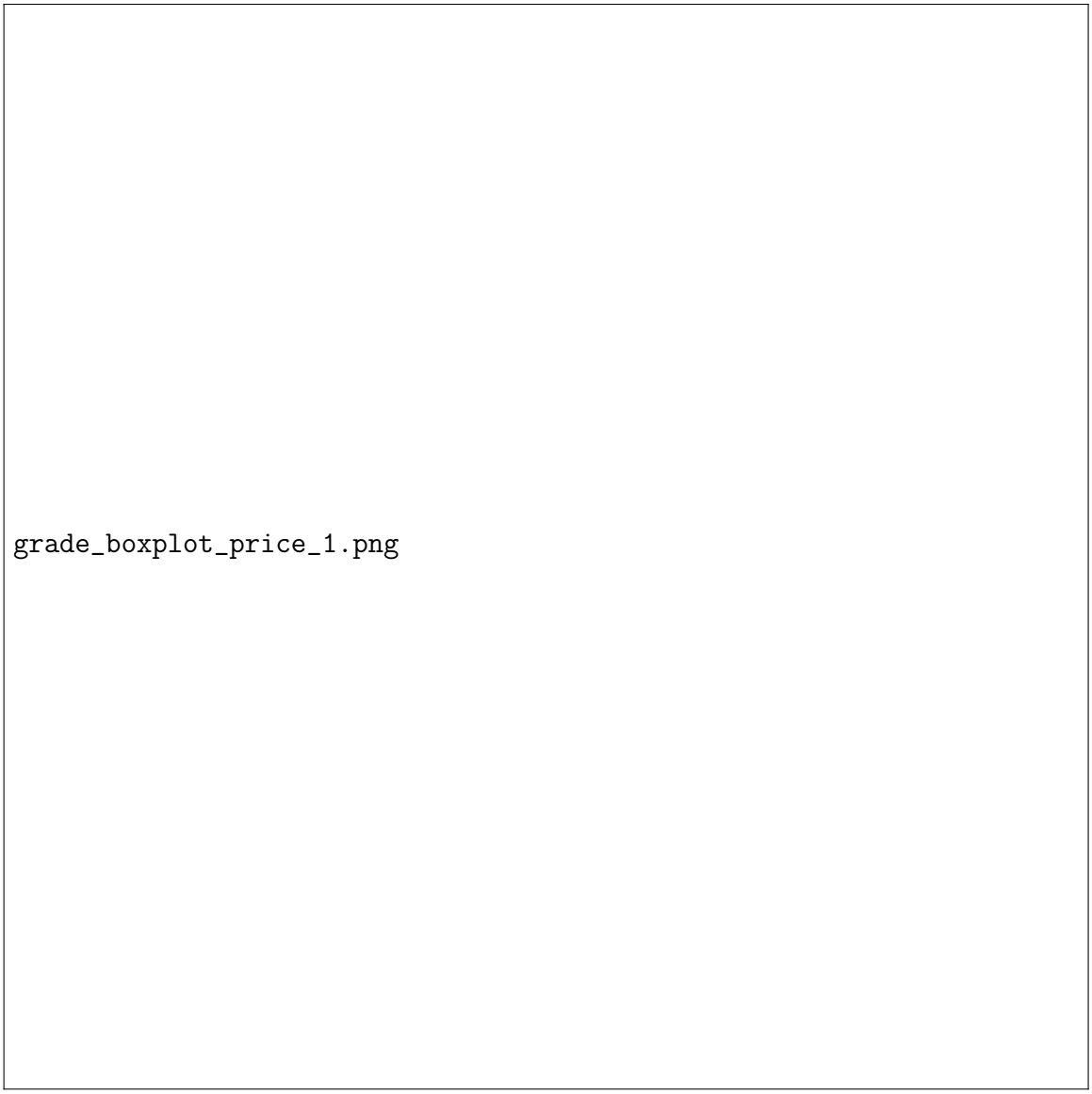

Figure 3: Price Boxplot

3.4 Feature Engineering

We apply feature engineering on the imputed data set. Note that some of these engineered features are used in certain methods, while some are not.

- Standardize columns *GBA* and *LANDAREA*
  It can be observed that excluding outliers, the histogram plot of these two variables are slightly right-skewed, but close to normal. Standardizing tends to make the training process well behaved because the numerical condition of the optimization problems is improved.
- Create column *RM* where $RM = BATHRM + HF\_BATHRM + ROOMS$
- Sales count this year/this month: number of sales made this year/ month
- Houses count within NBHD/Ward/CT/Quad: The number of houses allocated within its corresponding neighborhood/ward/census tract/quadrant

We also manage to get some useful information out of the *SALEYEAR* column. These additional features include:

- *SALEYEAR*: The year in which the property is sold, which is converted into numerical values
- *SALEMONTH*: The year in which the property is sold, which will then be converted into a categorical variable because we believe seasonality may have an effect on real estate pricing
- *SALEQUARTER*: The quarter in which the property is sold, which will then be converted into a categorical variable for the same reason as above
- The number of days from the sales date to today's date. We would like to see how the price behave given the most recent sales versus one that has been sold a while ago: *SALEDATE_TODATE* = Today's date (as April 15th) - *SALEDATE*
- How long does it take for a property to get sold after it is re-modeled: *RMDL_TO_SALEYEAR* = *SALEYEAR* - *YR_RMDL*
- How old is the property in years since the earliest time the main portion of the building was built: *PROPERTY_OLD_YEAR* = *SALEYEAR* - *AYB*

Other features include:

- We algo convert a couple more columns into categorical including *ZIPCODE.USECODE*
- *DIST* is the haversine distance (spherical distance) between the the expensive property and others. This is calculated using *LONGGITUDE* and *LATTITUDE* columns

- We re-assign an ordinal value score to different categories in *GRADE* and *CNDTN* columns. For example, an 'Average' value in *GRADE* will have a score of 0 and an 'Above Average' will have a score of 1
- Under *outliers*, we try to assign the label of 1 to those properties with *PRICE* exceeding 2 standard deviation from the population mean. The rest will have the label of 0
- We applied one hot encoding to all categorical variable
- Before finalizing our processed data and use it as input to our models, we drop the following columns name: *NATIONALGRID*, *ASSESSMENT_SUBNBHD*, *CENSUS_BLOCK*, *FULLADDRESS*

3.5 Outliers

The outliers that were identified in imputation section have been replaced (corrected), and will thus be treated as regular data points from this point forward. In this section, we discuss outliers that are found within the data, but cannot be replaced given the lack of field knowledge.
Firstly, data points with Id 17400 and 19136 are outliers, because their corresponding price is 250, which is clearly erroneous. It can also be observed that both of these points lie within the specified 5th fold, which explains why the RMLSE prediction error from this fold is much higher than other folds. Id 798 is also an outlier, since it has 0 rooms yet a relatively high price. From plotting data points we can clearly spot the outlier point corresponding to Id 7518, which has a price of over 20 Million.

We will handle these points by excluding them from the training data in the methods described later, but will be including them within the test data set for predictions.

4 Methodology

For this method, we attempt to fit the processed data set a series of models. We will then classify the observations in the test set using each of these models and present a final classification decision using a weighted voting procedure.

4.1 Bayesian Models

Many of the models we consider are based on Bayesian theory, in the sense that we can identify the posterior probability that an observation falls in class $k \in \{0, 1\}$. We need to determine the prior probability of class k (which is, in practice, the proportion of observations in the training set that belong to class k) and, more importantly, a class conditional density of our explanatory variables for class k. The following methods are concerned with the identification of such densities.

4.1.1 Model 1: Linear Discriminant Analysis

In linear discriminant analysis (LDA), we model the class densities with a multivariate Gaussian density with mean μ_k and variance matrix Σ. The key assumption in LDA is that every class has the same variance structure and, as such, the decision boundaries between the Gaussian centroids are linear.

4.1.2 Model 2: Quadratic Discriminant Analysis

Quadratic discriminant analysis (QDA) is similar to LDA in the sense that both models assume a Gaussian distribution for the class densities. However, QDA relaxes the assumption of constant variance by allowing the covariance structures for each class to vary, resulting in a less biased model albeit with higher variance.

Our LDA and QDA models fitted on this dataset exhibit a pecularity: although the bias for the QDA model is theoretically lower than it is for the LDA, we notice that the QDA training error for QDA exceeds the error for LDA. This is a rather rare occurence

4.1.3 Model 3: Naive Bayes

4.1.4 Model 2: No interactions with Customized k value

Observing the output given by gam.check of the previous model as seen in Appendix - Smoothing, we find that the k values for *LANDAREA* and *HousesInWARD* are too small. We also notice from the summary of the model that *USECODE* is an insignificant variable which should be dropped. These changes lead to the second formula attempted:

$$\begin{aligned} log(PRICE) \sim & s(BATHRM) + s(HF\_BATHRM, k = 5) + s(ROOMS, k = 13) \\ & + s(BEDRM) + s(AYB, k = 14) + s(YR\_RMDL, k = 13)... \\ & + HEAT + AC + ...WARD + QUADRANT \end{aligned} \tag{1}$$

4.1.5 Model 3: GAM with Interactions (Submitted on Kaggle)

After further adjusting the k values within smoothers, we find that the RMLSE value still fluctuates around 0.206 to 0.208. We now consider adding interaction terms. Due to memory limits in R, it is infeasible to incorporate interactions between every pair of features. An alternative is to sample a subset of all interactions and choose our interaction terms on a trial and error basis. Working on the Random Forest algorithm in parallel helped with the interaction term selection. SALEYEAR and BATHRM are found to have the highest importance under Random Forest, which implies a strong correlation between them and the housing price. From there, we find

that taking interaction terms that involve SALEYEAR and BATHRM significantly improved the GAM model performance. This resulted in the final model.

$$\begin{aligned} log(PRICE) \sim & s(BATHRM) + s(HF\_BATHRM, k = 5) + ...QUADRANT \\ & + ti(BATHRM, HF\_BATHRM) + ti(BATHRM, BEDRM) + te(RM, AYB) \\ & + ti(RM, YR\_RMDL) + te(RM, STORIES) + ti(GBA, STORIES) \\ & + ti(LANDAREA, GBA) + te(SALEYEAR, STORIES) \\ & + ti(SALEYEAR, GBA) + ti(SALEYEAR, RM) \\ & + ti(SALEYEAR, YR\_RMDL) + ti(SALEYEAR, LANDAREA) \end{aligned} \tag{2}$$

4.2 Comparison between models

	GCV	AIC	BIC	RMLSE
Model 1	0.04045	-11695.261	-8515.8394	0.20734
Model 2	0.04044	-11937.546	-8289.282	0.20689
Model 3	0.035466	-15855.9305	-11391.619	0.19856

It can be seen that among all 3 models, the last model has the lowest GCV, AIC, BIC and RMLSE. The GCV score is the minimised generalised cross-validation (GCV) score of the GAM fitted. Here GCV can be used to estimate prediction error. AIC and BIC are both penalized-likelihood criteria, except BIC penalizes complexity more heavily. A low GCV indicates lower prediction error and lower AIC/BIC indicates better quality/fit of the model. This aligns with the finding that Model 3 has the lowest RMLSE value.

5 Random Forests

5.1 Variables to Train

The final Random Forest selected is fitted on the processed data with the added engineered feature *RM*. In total, the data frame contains a total of 31 numerical and categorical predictive variables. We have attempted to fit the model with additional engineered features, however, doing such ended up with higher prediction errors. A possible explanation for this is that the Random Forest algorithm randomly selects samples of variables and attempt to construct many de-correlated decision trees. Since the engineered features are computed based on the original features (variables) in the data set, the constructed trees are likely going to be more correlated, thus more prone to overfitting which results in higher errors.

5.2 Tuning

Using the R package 'ranger', we tune the following hyper parameters:

- "num.trees": The number of trees in the forest
- "m.try":Number of variables to possibly split at in each node
- "min.node.size": Minimal node size

We first attempted to use the R caret package to help grid search hyper-parameters using 5 fold cross validation. However, this approach lead to R memory issues. To work around this, we created our own grid search function with for loops. This approach was very time consuming. Given that there are 3 hyper-paramaters to tune, time used goes up exponentially as the grid size increases, allowing us only able to test a few hyperparameter combinations.

As a result, we split the entire dataset into a train set and a test set, and selected 5 hyperparameter combinations that resulted in the lowest RMLSE on the test set. We then evaluated the overall RMLSE prediction errors for all 5 of the selected models, and determined the best model among them.
The final model chosen has m.try = 15, num.trees = 800, min.node.size = 2, which returns a RMLSE of approximately 0.180.

5.3 Importance of Variables

The following features have the highest importance under the final model:

	SALEDATE	LONGITUDE	WARD	GBA	GRADE	BATHRM
Smoothing	6278.399	4090.640	1519.605	1182.46	726.939263	556.440858

Variables with high importance have a significant impact on predicting the outcome. It is understandable that *SALEDATE* has such high importance as it was also demonstrated to be a driving predictor under the Smoothing Method. For a more visual representation of the feature importance, please see Figure 4.

6 Boosting

6.1 Apply Boosting on the data

We implemented gradient boosting using lightGBM python package. Unlike any other tree based algorithms where trees are grown in level-wise, lightGBM grows the trees leaf-wised. It selects the leaf with the max delta in the loss function to grow. To grow the same leaf, level oriented trees growing can take more time and memory. Boosting model using the optimal hyper parameters is trained and validated through 5 folds cross validation. The input data frame consists of 165 features (numerical and categorical combined). The algorithm is set to exclude *(*Id), *(*PRICe) and *(*fold) during

Figure 4: Random Forest Feature Importance

training and validation. Our boosting algorithms produces an RMSE of 0.16746 as shown on Kaggle.

6.2 Hyper Parameters Tuning

The following hyper parameters are tuned:

- learning_rate: The learning rate used to train the data. The smaller the rate is the slower the longer it takes to learn. We set the search range for this parameter to be from 0.005 to 0.025

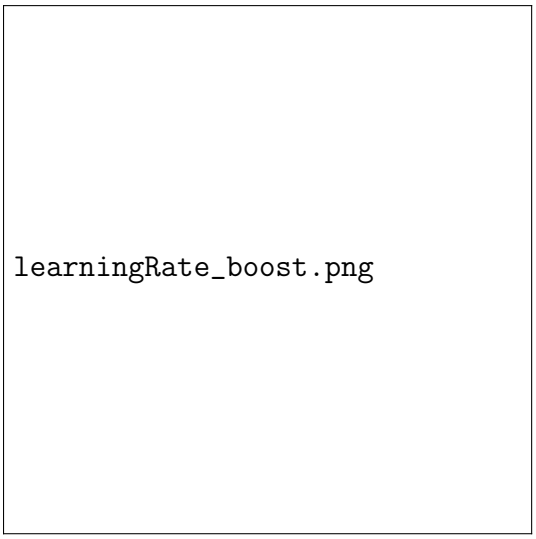

Figure 5: Learning Rate Distribution

- max_depth: The maximum depth of tree. This parameter is used to offset over fitting.
- min_data_in_leaf: The minimum number of the records a leaf may have. This is also used to offset over fitting. The search range for this is from 1 to 22
- bagging_fraction: The fraction of data to be used for each iteration. This parameter is searched in a range from 0.05 to 0.4
- feature_fraction: The fraction of features to be used for each iteration. This parameter is searched in a range from 0.10 to 0.7
- num_boost_round: The number of boosting iterations for each fold. We set this to 10000 iterations per fold
- early_stopping_round: If the valid result does not improve after a certain number of training rounds, this parameter will prevent excessive training. We set this to 200

6.3 Selection Process

A 5-fold cross validation randomized search is used to tune for the optimal hyper parameters for the model. We searched through 120 different random combinations from the parameters grid, each of which is applied to data and the corresponding rmse is stored. The set of hyper parameters which produces the smallest rmse is selected to be used for the final global boosting model. Below is the set of hyper parameters we used:

- num_leaves: 256

- min_data_in_leaf: 20
- learning_rate: 0.0055
- feature_fraction: 0.4508163265306123
- bagging_fraction: 0.2952525252525253

With these hyper parameters, our boosting model achieves an RMSE of 0.16746. Below is the distribution of the boosting predictions:

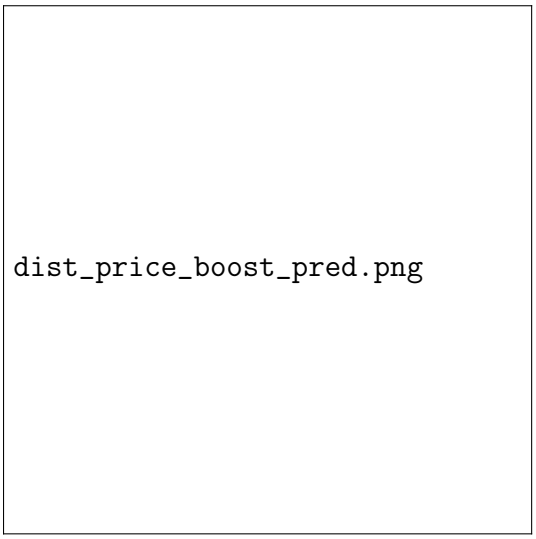

Figure 6: Distribution of Boosting Predictions

6.4 Importance variable

The feature importance is determined by the the total gains of splits which use the feature.
Interestingly enough, we observed that features that have relatively high correlation to the *PRICE* contributed the most to the prediction of boosting algorithms

7 Other Methods

One other that we try is that we take avaerage of the predictions from our random forest and boosting algorithms, which produces the RMSE of 1.7162 as showned in Kaggle.

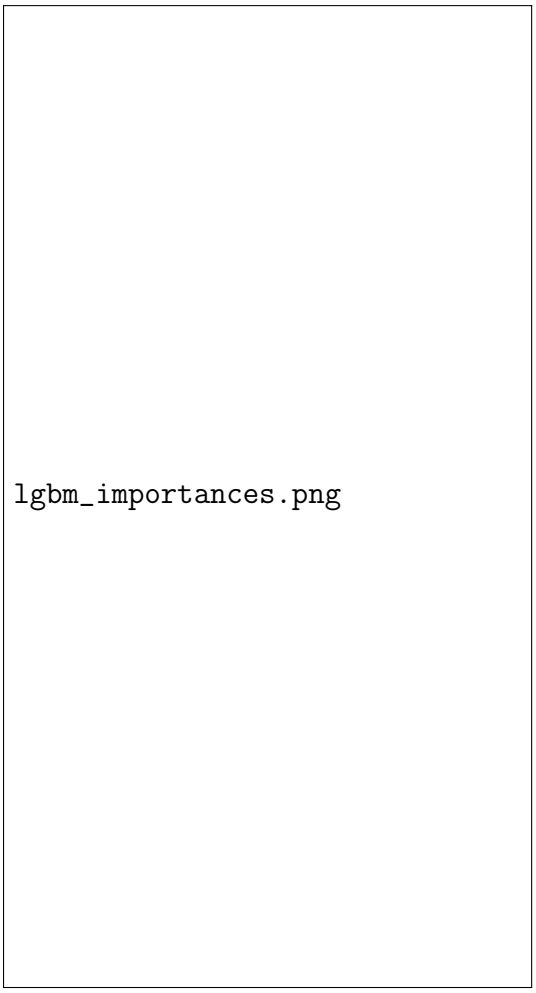

Figure 7: Boosting Feature Importance

8 Statistical Conclusions

	Prediction Error (RMSE)	Computational expensive	Ease of Implement
Smoothing	0.198		✓
Random Forest	1.80		✓
Boosting	1.6746	✓	

Overall, the boosting algorithm clearly outperforms other models by a good margin. Despite the fact that boosting takes a long time to train due to the larger number of tuning parameters, we believe that it is the best candidate given the prediction errors and sensitivity to over fitting. Generally speaking, one down side of boosted trees is that it is a lot harder and computationally expensive to tune than Random Forest. This is due to its larger number of hyper parameters as compared to random forest. Since boosting performs optimization in the objective function base rather in parameter space, the way it sequentially learns makes boosting a better candidate

than Random Forest. With random forest, trees are trained independently from each other using the fundamental of boostrap and bagging. This makes random forest less sensitive to overfitting than boosting. For data including categorical variables with different number of levels like what we have in this data, random forests might be biased in favor of those attributes with more levels. Hence, the variable importance scores from random forest are not reliable based. The feature selection process for smoothing appears to be much more arbitrary compared to the other methods.In other words, Random Forest and Boosting algorithms have a more systematic and optimized way of selecting important features, which makes up for their increased computational time.

9 Future Work

9.1 Smoothing Methods

As previously mentioned in 4.1.3, interaction terms involving *SALEYEAR* can improve the GAM model performance. After the Kaggle submission deadline had passed, we trained the GAM model with the same one-variable smooth terms and categorical variables as Model 3, but added all interaction terms that include *SALEYEAR*. This appeared to decrease the GCV, AIC, BIC and RMLSE values even further than Model 3. A summary of this model can be found below. Details of the model can be found in the "Further Worked Done" section in Appendix - Smoothing.

	GCV	AIC	BIC	RMLSE
Smoothing	0.031456	-19652.58	-14255.73	0.1876124

9.2 GAM Modelling

There are other paramaters to be tuned within the GAM model under the mgcv package. For instance the 'bs' value which indicates which spline smoothers are used and the gamma value which adjusts the smoothness of the model. Should there extra time, we would like to explore more of these options.

10 Contribution

	Hieu Nguyen	Alice Li
Smoothing	50%	50%
Random Forest	50%	50%
Boosting	50%	50%