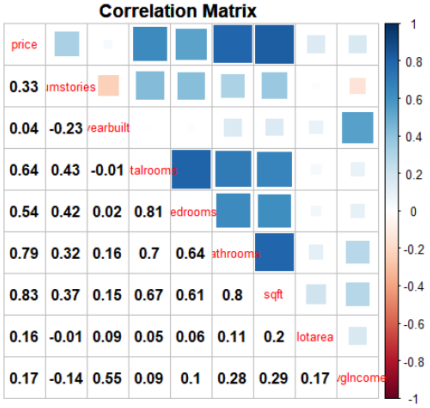
Technical Report

1. Introduction / EDA: The goal of the project is to predict the price of a house as well as understand what the most important factors in a house value are. Two datasets were provided: the train dataset with 1200 observations and 17 features including `price,` and the test dataset with 600 observations and 17 features (N/A for `price`). I performed Exploratory Data Analysis to learn more information about the data and preprocess it based on the analysis. Below is a table summary of the final data structure with the number of unique values per variable per data set.

Variable	Type	Description	Train	Test
id	Index	Unique property identifier	1400	600
price	Numeric	Price of the home -- **RESPONSE**	1400	1 (NA)
numstories	Numeric	Number of stories	8	9
yearbuilt	Numeric	The year the home was built	139	125
totalrooms	Numeric	Number of rooms in the house	18	5
bedrooms	Numeric	Number of bedrooms in the house	9	8
bathrooms	Numeric	Number of bathrooms in the house	18	14
~~fireplaces~~	~~Numeric~~	~~Number of fireplaces in the house~~	~~7~~	~~5~~
sqft	Numeric	Square footage of the house	1018	506
lotarea	Numeric	Lot area in square footage	1103	485
AvgIncome	Numeric	The average household Income per zipcode	45	43
desc	Factor	Description of home	5	5
exteriorfinish	Factor	The exterior finish of the home	6	6
rooftype	Factor	The material of the roof	4	4
basement	Factor	Indicator of if the home has a basement	2	2
state	Factor	The state the house is located in	2	2
~~zipcode~~	~~Factor~~	~~The zipcode the house is in~~	~~45~~	~~43~~

I set `id` feature as an index because it gives no information about the price. `fireplaces` is the only column that has a lot of *missing values* in both datasets: 687/1400 in the training dataset, and 313/600 in the testing dataset. I considered a couple of ways of dealing with missing data. First, we can fill in the missing with a default value. For this type of problem, the default value is not obvious because the average number of fireplaces or a `0` default value might be misleading for the models especially given so many missings. Second, we can delete missing observations

from the dataset. However, the number of observations to remove is too large; we would have had deleted almost half of the available data. Therefore, I have decided to NOT use the `fireplaces` column to make predictions because I want to have all available observations. Interestingly enough, all the observations with missing `fireplaces` data are from `VA` state! There were no *variables with near-zero variance.* I checked the number of *unique values* for each variable. I decided to remove `zipcode` because it corresponds to `AvgIncome`. I left `AvgIncome` as a numeric because the amount of money makes sense to keep as a numeric. I kept `totalrooms,` ` numstories `, `bedrooms,` and `bathrooms` numeric so that the models can support new data with different values for these variables. With visualizations, I found that there are some outliers in the continuous variables - `price`, `AvgIncome`, `lotarea`, `sqft` - but I decided to keep them to have more data available for training. I also plotted the distributions for all the variables. Factor counts for `desc`, `exteriorfinish`, `numstories`, and `rooftype` are highly uneven. Finally, the correlation matrix revealed that `price` is highly positively correlated with `sqft`, `bathrooms,` `totalrooms,` and `bedrooms.` These 4 variables are highly positively correlated with each other. The correlation plot also shows that the newer the building, the fewer stories it has.



2. Methods Overview / Details: For the models discussed, I trained them on the train subset of the train data set, and tested on the test subset of the original train dataset. Test MSE was used as a performance metric. Since we have a fair number of factor variables with uneven distributions, I performed ***stratified sampling*** with 80/20 split on train data set to make sure that all the factor levels are represented in the train and test equally. Where appropriate, I performed

method	test_MSE
RF	6047291110
Boosted	7180889097
Bagged	10532220607
PLS	10689650924
Lasso	10697254072
PCR	10742195148
LM	10745202369
LM with 7	10933707494
Ridge	10978949422
LM with 3	12154330974
Poly	17164105421
Tree	21359816377
Spline	23214703112
KNN	36251762340
Baseline	79598514171

Cross-Validation to choose the best tuning parameters or a number of components / variables to use. For CV, I used all of the original training data (1400 observations). I've tried linear models first, and then went onto non-linear to compare. The first model I have tried is a linear model with all the variables since it is the simplest model we can do. Next, I performed Best Subset Selection, Forward Selection, and Backward Selection to identify the most important variables. When graphing Test MSE, I found that models with either 3 or 7 variables were best. Interestingly enough, all three methods resulted in the same set of variables for subsets of both 3 and 7 sizes. The three most important variables were `sqft`, `bathrooms`, and `stateVA`. For model with 7 predictors, `desc`, `exteriorfinish`, rooftype`, and ` avgIncome` were added. Then, I trained linear models with only those 3 and 7 predictors. After that, I trained Ridge and Lasso with all the features to penalize complexity and see if regularization improves the results. The next step was to use PCR and PLS to uncorrelate the features and reduce the number of predictors used. The best number of components was 20 for PCR and 6 for PLS. After that, I went onto non-linear methods: polynomials and splines on `sqrt` and `bathrooms` since EDA showed them to be the most important variables. The next set of models is more complex non-linear models: tree methods with the number of trees = 1000. I've trained a Tree, Bagged Trees, Random Forest (mtry=7 via CV) and Boosted Trees (lambda=0.01 and interaction.depth=4 via tuning). For the tree models, `sqft` and `bathrooms` were the most important variables. The first tree split was always on `sqft`. Finally, try one of the clustering methods, I did KNN regression with `k=5` found via CV.

3. Summary of Results: Random Forest performed the best with Boosted Trees finishing second. Their test MSEs were fairly close to each other. I think that RF did better than Boosted because the latter might be overfit to the data since the model is fitted on the residuals.

Additionally, the graph of test errors vs lambda / interaction depth combination was very cyclical with an overall increasing trend; the best set of tuning parameters was not obvious. I chose the ones that give the absolute lowest test MSE. The next in line was Bagged Trees method that is close to the following 6 linear models (please see the table). PLS, Lasso, and PCR performed well which suggests that uncorrelating the features was an important step in getting good predictions. The linear model with 7 variables performed better than the linear model with only 3 variables which suggests that more variables than just `sqft`, `bathrooms`, and `state` were important. Overall, all linear models were very close to each other! Out of non-linear models, RF, Boosted Trees, and Bagged Trees did best because they are more complex than Polynomial and Splines with 2 features used. KNN was the worst. All the methods were better than a baseline averaging.

4. Conclusions / Takeaways: Overall, I think that it is safe to trust the Random Forest model predictions, and I've tuned `mtry` via CV to ensure the best performance. However, `fireplaces` and `zipcode` was excluded at the very beginning, so I would not rely on that information when making a decision. What else was challenging is that some of the variables had only a few observations for different levels. For example, `desc` `MOBILE HOME` (1), `exteriorfinish` `Log` (3) and `Concrete` (4) as well as many `zipcode` / `AvgIncome` levels. I would not completely trust predictions for those because we did not have enough data on them to start with. I go into more details on that in the non-technical report to save space here. Even though I considered many models, something else that can be explored further is creating models based on subsetting. For example, I found from the EDA that all the observations that have missing `fireplaces` values are from `VA` state. Therefore, we could train different models based on the state. In this case, we would have the `fireplaces` feature in the data for `PA,` and remove that column when training the models for the data with the state `VA.`