Predicting House Prices Using Textual Information

Problem Statement

There are already many tutorials/sample projects about predicting house prices using a variety of machine learning regression models (https://www.kaggle.com/c/home-data-for-ml-course). However, many of them use the famous Ames or Boston house datasets, which can be impractical for real-world house hunters to apply to their price prediction problems for the following reasons. First, the Ames dataset has many quality or condition features that are impossible for real-world house hunters to gather. Nowadays potential buyers mainly rely on websites like Zillow.com and Redfin.com to get basic information about a listing, which is very limited compared to the features in the Ames dataset. The Boston dataset, on the other hand, only provides aggregate features like average, median, proportion, etc., of the neighborhood. Obviously it is not fair to judge an individual house's price just based on the neighborhood aggregate features. Second, those datasets do not contain information about listing prices. It is natural for a potential buyer to ask "how much should I add/deduct from the listing price as my offer price" when a new listing shows up.

The goal of this project is to address these two problems. A real-world house dataset was scraped from Zillow.com, which contains only ~30 quantitative/factual features devoid of qualitative/condition features like in the Ames dataset or neighborhood aggregate features like in the Boston dataset. A main hypothesis of this project is that the lack of quality/condition information on the houses can be supplemented by the raw description accompanied the listing, i.e., the "Overview" section on the listing webpage on Zillow.com. These descriptions may provide "clues" as to the quality/condition of the houses. For example, "luxurious" or "many updates" suggests a higher quality and hence sale price, while "priced-to-sell" or "motivated seller" indicates a lower quality and hence sale price, compared to "average" houses with the same metrics such as size or year built in the same neighborhood. The dataset focuses on the city where I live, Cary, NC, out of my own interest. It is a relatively small town, and I envision any difference in neighborhood characteristics should be captured by the zip code and the assigned school ratings, which are included in the features scraped from Zillow.com.

Data Source

All the data of each sold property was scraped from Zillow.com. First, lists of urls were scraped from Zillow.com searching for 'Cary, NC' and filtering for 'Sold' properties. This was done on Feb. 18, 2022 and the resultant list contains properties sold 4 months prior to the time when the scraping was performed (Nov. 2021 ~ Feb. 2022). Second, data on each property's Zillow.com url was scraped, including the following features: status, number of bedrooms, number of bathrooms, living area, address, sold price, sold date, description, listed by, listing data, listing price, house type, year built, parking space, lot size, most recent three years tax assessment, elementary, middle, high school assignments and school ratings. This serves as the training set. Later on April 18, 2022, I scraped an additional dataset consisting of properties sold from Feb. 18 to April 18. This set will be used as the test set.

Methodology

1. Data cleaning and feature engineering

This step includes:

- Drop the ‘status’ column as it is all ‘Sold’.
- Convert the ‘soldDate’ and ‘listingDate’ columns to the datetime type.
- Remove data points with a generic description “(address) is a (type of home) that contains (area) sq ft and was built in (year). It contains (number of) bathrooms. This home last sold for $(sale price) in (sale date). The Zestimate for this house is $(price). The Rent Zestimate for this home is $(price)/mo.” This kind of generic description is of no use to extract textual information to build the models and is often associated with lot/land sales.
- Consolidate the ‘houseType’ column to have only three categories: Single Family Residence, Townhouse, and Condo.
- Fix the ‘yearBuilt’ column. Some very new houses did not have this information and were filled with wrong information when the data was scraped.
- Consolidate the ‘parking’ column to have only numeric values indicating the number of parking spaces.
- The ‘lotSize’ column in the original data frame had data in two different units: acres or sqft. Convert numbers in acres to sqft.
- Impute missing data. The ‘bedroom’ and ‘parking’ columns had missing data. Impute from ‘living Area’ and ‘houseType’. For missing values in ‘lotSize’, fill in 0 and add an additional categorical variable to indicate missingness.
- Extract zip code from address.
- Create categorical variables for data points whose listing agent are either Zillow, Redfin, Opendoor, or Offerpad – the major iBuyers in business, or Mark Spain, a local flipper.
- Categorize data points based on the difference between the listing price and the sale price. In the training dataset, about 10% of the data points did not have listing prices and were removed.
- One hot encode all categorical string variables with the first column dropped after encoding.

Some of the features were only used for exploratory data analysis. The columns that were kept to build the regression models are listed below. They represent the “factual data” in Figure 1.

Column name	**Explanation**
bedroom	Number of bedrooms
bathroom	Number of bathrooms
livingArea	Sqft of the living area
listingPrice	Price the house was listed
houseType	Type of the house (Single house residence, Townhouse, Condo)

yearBuilt	Year the house was built
lotSize	Size of the lot in sqft
parking	Number of parking spaces
elementaryRating	Rating of the assigned elementary school (1~20)
middleRating	Rating of the assigned middle school (1~10)
highRating	Rating of the assigned high school (1~10)
zip	Zipcode of the address (27511, 27513, 27518, 27519, 27523, 27560, 27607)

2. Natural Language Processing (NLP)

This is also part of the feature engineering process. Raw texts from the 'description' column were processed (converting all characters to lowercase, removing special characters, fixing typos and abbreviations, removing stopwords, and lemmatizing) to build a bag-of-words (BoW) representation and a term frequency-inverse document frequency (TF-IDF) representation focusing only on unigrams. Additionally, a doc2vec (d2v) model was also used to represent each description as a vector (explained below). Without going through the NLP steps above, raw texts from the 'description' column were used to extract length information on word count, character count, sentence count, average sentence length, and average word length. Together they represent the "textual data" in Figure 1.

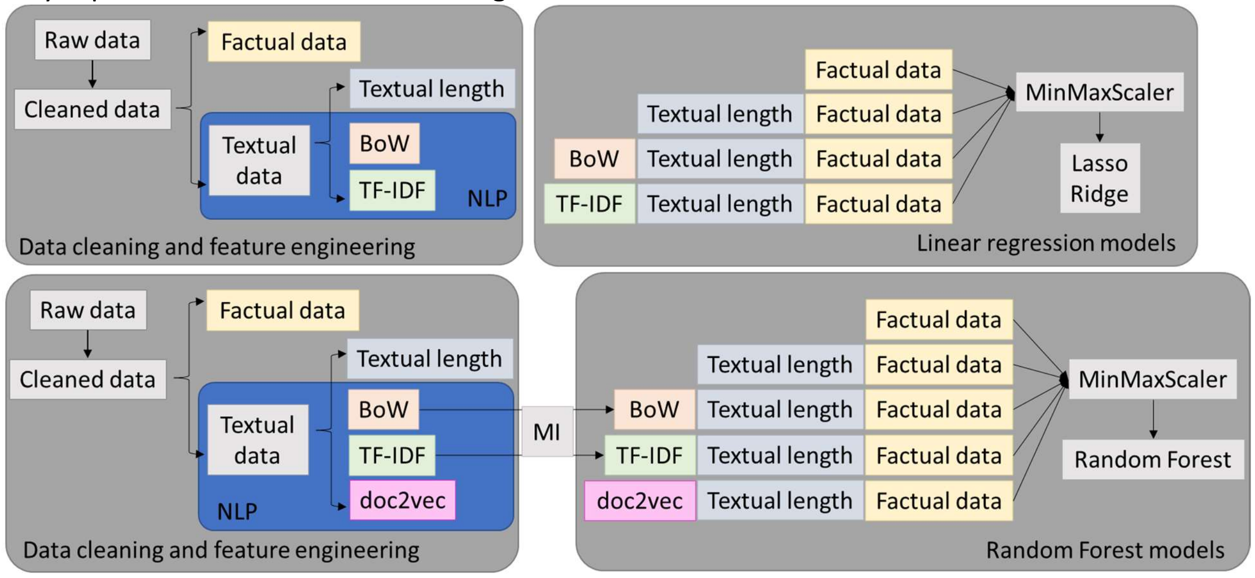

Figure 1. Methodology flow chart.

BoW and TF-IDF are two related basic NLP models for representing texts as numerical vectors. BoW simply counts how many times a word appears in a document. A collection of documents is called a corpus. A collection of words is called a vocabulary. After counting a corpus-vocabulary matrix can represent the whole corpus with each row presenting a document vector, each column representing a word vector, and each entry represent the count of the word occurring in the document. TF-IDF builds upon BoW to reflect how important a word is to a document in a corpus. In a TF-IDF matrix the count of the word occurring in the document is normalized by the count of that word occurring in the whole corpus (term frequency) and the number of documents in the corpus that contain that word (inverse document frequency).

Thus, TF-IDF makes rare words more prominent and effectively ignores common words. While BoW and TF-IDF are popular NLP models due to their simplicity, the main drawbacks include high dimensional and sparse data, and lack of semantic and contextual meaning. D2v is an extension to the word2vec (w2v) model, which learns to project words into a latent space where words similar in meaning have vectors that lie close to each other in space. D2v learns to represent each document as a vector such that similar documents will lie close together in space.

3. Regression models
For the regression model to predict the sale prices, I used linear regression with regularization such as Lasso and Ridge as well as Random Forest regression. They were chosen because of their easy interpretability (linear regression) or robustness (Random Forest). For linear regression models with Lasso and Ridge, all data was scaled by a MinMaxScaler, and then mean squared error (MSE) of the test dataset was used as a metric for the model performance. The comparison was made between using factual data only, factual data concatenated with textual length data, factual data concatenated with textual length data and BoW or TF-IDF representation (Figure 1 top). Because Lasso is a feature selection tool suitable for high dimensional data, I did not perform any dimensionality reduction step. For Random Forest regression models, because of the high dimensionality and sparsity of the BoW and TF-IDF matrices, an additional step to reduce the dimensionality of the data was needed. Here I chose mutual information (MI) to select features (words) that the target (sale price) is more dependent on. MI is a measure of dependence or "mutual dependence" between two random variables. It measures the average reduction in uncertainty about x that results from learning the value of y; in other words, the average amount of information that x conveys about y. Alternatively, a lower dimensional matrix was built using the d2v model to represent the raw descriptions (Figure 1 bottom).

Evaluation and Final Results

1. Exploratory data analysis
Initially I was concerned about the high rise of real estate properties, despite the short period of time frame, would require detrending of the data. However, the median sale price remained stationary during the period of Nov. 2021 to Feb. 2022 (training set, Figure 2 left) and Feb. 2022, to Apr. 2022 (test set, Figure 2 right), therefore there is no need to detrend the data.

Figure 3 shows the distribution of sale prices for the training and test datasets. As expected, both exhibit right skewed distributions. The test dataset also appears to have a second mode in the higher price region.

Sale-to-list ratio is a metric looking at the final sale price divided by the listing price expressed as a percentage. Figure 4 shows my local market has been very hot. A majority of the houses were sold above listing price, even more so in most recent months (Figure 4 right).

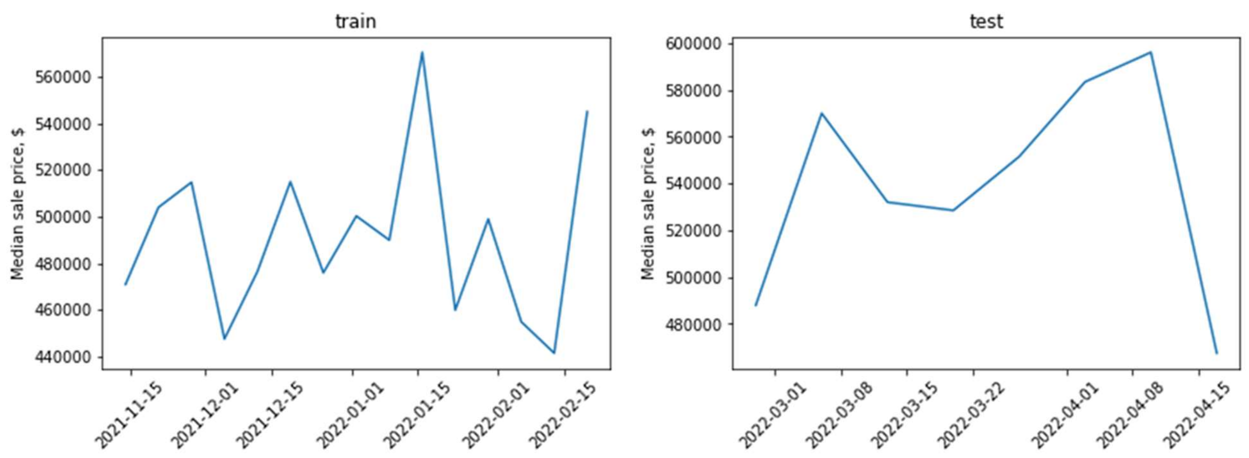

Figure 2. Median sale price over the period of the training set (left) and test set (right).

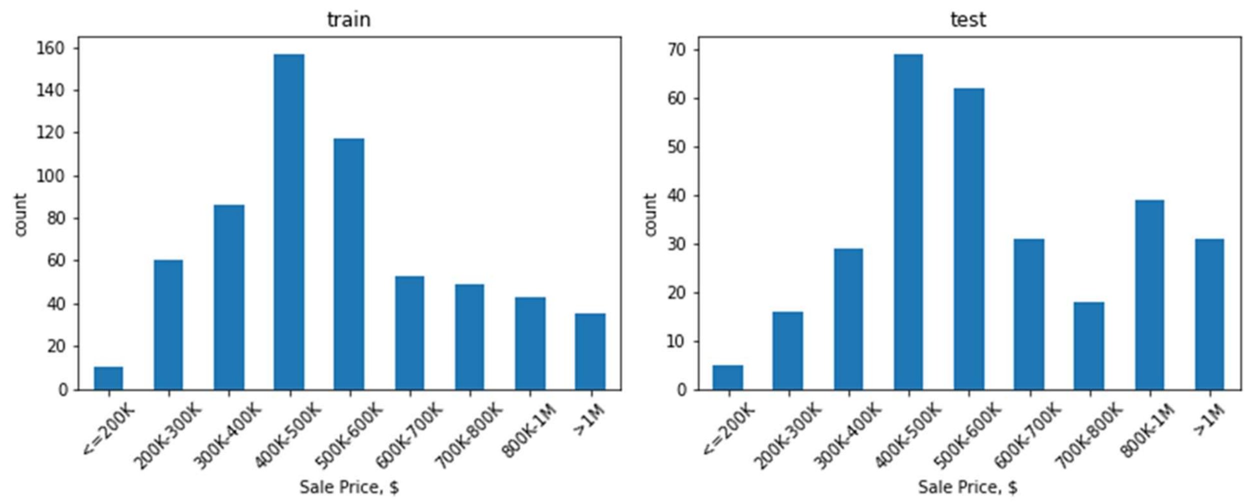

Figure 3. Sale price distribution in the training and test datasets.

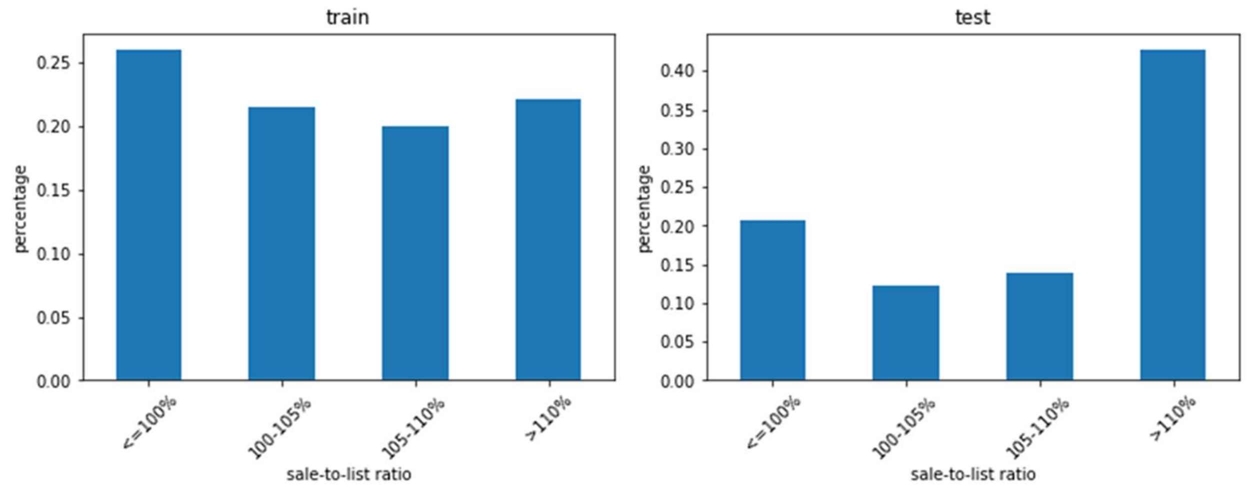

Figure 4. Distribution of the sale-to-list ratio in the training and test datasets.

The analysis on textual length shows mostly similar density distributions between the houses that were sold higher than the listing prices and the ones that were not, in both the training and test datasets, except for the sentence count (sentCount) and average sentence length (avgSentLength) measurements, which appear to be higher for the houses that were sold higher than the listing prices. This is an interesting observation, suggesting that houses that were sold higher are associated with descriptions with longer and more sentences.

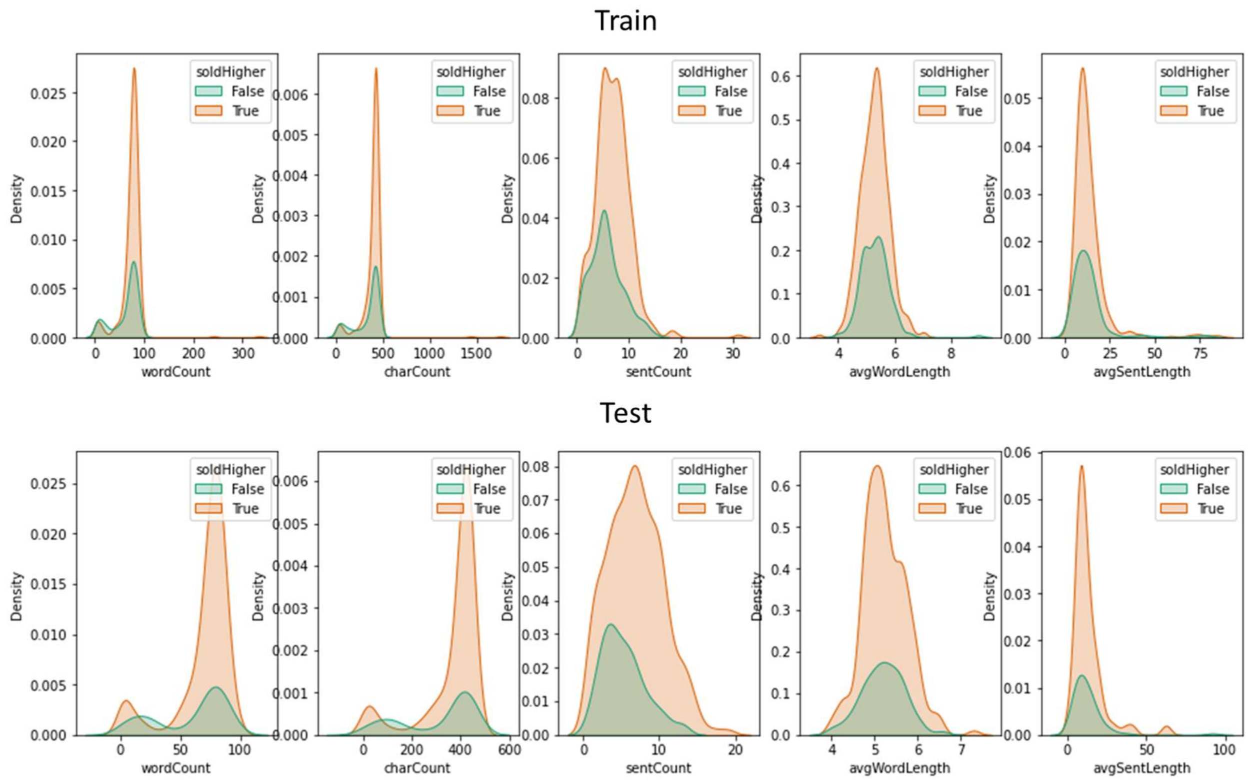

Figure 5. Length analysis in word count (wordCount), character count (charCount), sentence count (sentCount), average word length (avgWordLength), average sentence length (avgSentLength) of the raw description texts, comparing between the houses that were sold higher than listing prices and the ones that were not.

The word cloud generated from the BoW model shows the most popular words in the training dataset (Figure 6). Many of them are nouns describing parts of the house, like “kitchen”, “bath”, and “dining”. Some are qualitative adjectives, such as “new”, “large”, and “perfect”. Some are related to the location, for example, “cary”, “rtp”, and “downtown”.

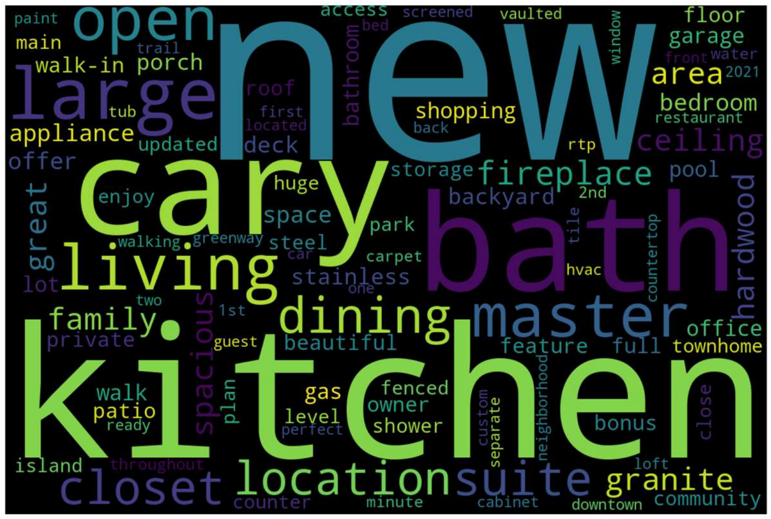

Figure 6. Word cloud of the most frequent 100 words in the training data set.

2. Linear regression models for sale price prediction

After the final cleanup, the factual data has 19 features. The textual information engineered from the raw description texts includes the length data of 5 features, and the BoW and TF-IDF matrices each having a vocabulary of 1531 words. The training set has 544 data points and the test set has 267 data points. As outlined in Figure 1, different inputs were subjected first to MinMaxScaler and then linear regression with regularization (Ridge or Lasso). The fitted model using the training data was then used to predict the test data, and the performance of the model was measured by the MSE of the test set, shown below. The results show that Lasso performed better than Ridge using all kinds of inputs. Adding textual length data improved only the Ridge model not the Lasso model. Further adding BoW or TF-IDF representation caused a huge deterioration in the performance of the Ridge model, due to the high dimensionality of the data and the fact that Ridge cannot perform feature selection. However, most importantly, including BoW or TF-IDF in the input improved the Lasso model about 9%.

Input data	**Best regularization for Ridge**	**Test error (MSE)**
factual data	alpha=0.1	4615442884
factual data + textual length	alpha=0.1	4464423559
factual data + textual length + BoW	alpha=0.1	33744548197
factual data + textual length + TF-IDF	alpha=0.1	31858918802

Input data	**Best regularization for Lasso**	**Test error (MSE)**
factual data	alpha=1400	4383021015
factual data + textual length	alpha=1400	4383020955
factual data + textual length + BoW	alpha=400	4027687704
factual data + textual length + TF-IDF	alpha=200	4077083615

Linear regression coefficients represent the magnitude and direction of the correlations between each feature and the target. As expected, the listing price has the largest coefficient in all of the models. Figure 7 left shows the non-zero coefficients for the Lasso model using only the factual data. To avoid dwarfing all other coefficients, the 'listingPrice' coefficient was not included in subsequent plots. When BoW was included in the input, Lasso picked up a few words that are positively correlated with the sale price (Figure 7 middle, features with coefficients less than 10000 in magnitude were excluded from the plot.). Among them, "extensive" has the largest magnitude. In the context of real estate listing, "extensive" is often used to describe flooring, molding, landscaping, trim work, updates, etc. It makes sense that a house being described as "extensive" can indeed have a higher quality. Other interesting words picked out by Lasso include "screen" and "porch" (suggesting a screened porch is a desirable feature of a house), "price" (as in "priced to sell", "price reduced", or "new price", negatively correlated with the sale price). Other words did not make much sense, like "sq", "stair", "2015", which emphasizes the need for representing a document with more sophisticated models as well as careful preprocessing of raw texts in NLP.

Using TF-IDF representation in the input also resulted in similar features being picked out by Lasso (Figure 7 right, features with coefficients less than 20000 in magnitude were excluded from the plot.).

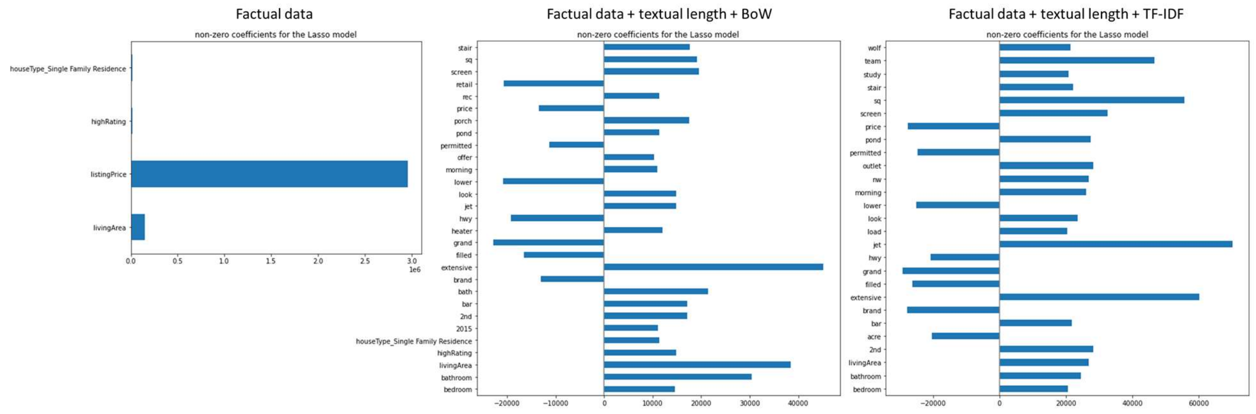

Figure 7. Coefficients of the Lasso models using different inputs. 'listingPrice' coefficient was excluded in the middle and right plots to avoid dwarfing other coefficients. In the middle plot, only coefficients with magnitude greater than 10000 are shown. In the right plot, only coefficients with magnitude greater than 20000 are shown.

3. Random Forest regression models for sale price prediction

For Random Forest models, I used a similar work flow, tuning for the best parameter with the training data and then using the best model to predict the test data. This time, simply adding the textual length data improved the performance of the model. However, with BoW or TF-IDF representation, not only the performance is worse than just using the factual data only, the running time is significantly longer due to the high dimensionality of the data. To reduce dimensionality, I used MI to select most relevant features (words). With BoW, I found a vocabulary size of 70 is the best and produced a lower test error than without BoW. With TF-IDF, the best vocabulary size was also found to be 70 but there was no improvement using the reduced dimension matrix. Another way of reducing the dimensionality of the data is to build a low dimensional matrix to begin with. To do that, I used the d2v model, representing each description with a vector of an assigned size. I found the vector size of 30 resulted in the best performance of the subsequent Random Forest model; however, the test error did not decrease from the input data without the vector representation. Note that due to the time and resource limit, for all the Random Forest models I did not do a full grid search for the best parameters and the models could certainly benefit from doing that.

Input data	**Best parameter for Random Forest**	**Test error (MSE)**
factual data	max_leaf_nodes=170	5792442276
factual data + textual length	max_leaf_nodes=180	4505426510
factual data + textual length + BoW	max_leaf_nodes=120	6174061229
factual data + textual length + TF-IDF	max_leaf_nodes=230	6106529351
factual data + textual length + BoW (reduced dimension)	best 70 vocabulary with max_leaf_nodes=120	4378652422
factual data + textual length + TF-IDF (reduced dimension)	best 70 vocabulary with max_leaf_nodes=230	4552982491
factual data + textual length + d2v	vector_size=30 with max_leaf_nodes=250	5031805707

To gain insights into the importance of each features in the Random Forest models, I used permutation importance, which calculates the increase in MSE (estimated with out-of-bag samples) as a result of features being permuted (values randomly shuffled). This procedure breaks the relationship between the feature and the target, thus the increase in MSE is indicative of how much the model depends on the feature. The higher the number, the more important the feature. Again, as expected, the 'listingPrice' feature dwarfed all other features (Figure 8). Interestingly, in the model with the textual length data included in the input, average sentence length (avgSentLength) was the second most important feature (Figure 8 right), in line with the observation that houses that were sold higher than the listing price tend to have higher sentence count and higher average sentence length (Figure 5).

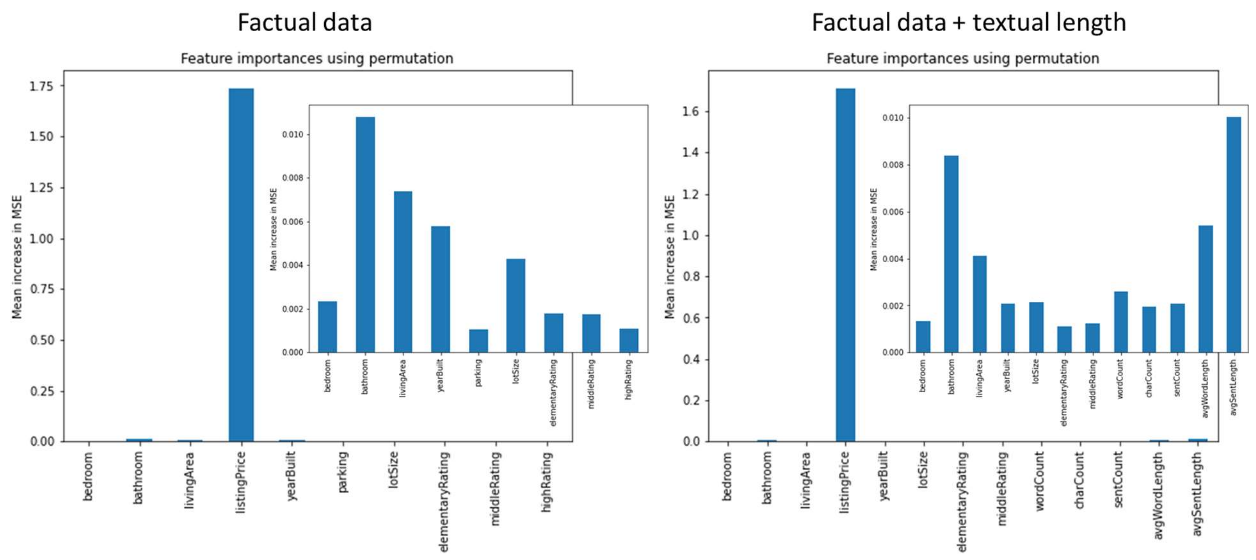

Figure 8. Permutation importance of features in the Random Forest models using different inputs.

Discussion and conclusions

In this project, I employed different machine learning techniques to build regression models to predict house sale prices. The novelty of my project lies in the inclusion of property descriptions as the predictors of the response. For machine learning models to understand the textual information, the information must be presented as numbers to the computers. In this project I used simple BoW and TF-IDF models as well as analysis on the length of words and sentences to represent the textual information. In several occasions, models including the textual information outperformed models without. The best MSE of my models is about 4×10^9, which means the RMSE is about 63000. For a 3-million-dollar house, this is only a 2% error. But for a 300K dollar house, this constitutes a 20% error. While at this stage my models certainly cannot compete with Zillow’s Zestimate, this project serves as a proof of concept for using textual information to predict house prices.

The main hurdle of my models is to preserve the semantic and contextual meanings of the words in a paragraph, which BoW and TF-IDF ignore. A d2v embedding similar to w2v embedding was used to build a low dimensionality document embedding, although it did not help improve the Random Forest model. In the future more sophisticated word embedding models such as GloVe or BERT could be used to detect more detailed nuances in those description texts.