Machine Learning for Price Prediction

Team Principal Miners

Chiraag Kala, Sarah Qin, Siddhartha Khetawat, Srihari Seshadri and Terry Wang

Agenda

Problem Statement

Exploratory Data Analysis

- Variable Exploration, Feature Engineering, Correlations, Clustering

Initial ML Models

Data Manipulation

- Outlier Removal, Transformations, Standardization, SMOTE, PCA

Final ML Models

NLP Analysis

Problem Statement

Background

- Airbnb's accommodation marketplace provides access to 5+ million unique places to stay in more than 81,000 cities and 191 countries
- Marketplace companies such as Airbnb rely on commissions from their product offerings to generate profit. An optimized pricing model would help in stabilizing revenue growth.

Problem

- On Airbnb, hosts determine the nightly rate for their listing, but often lack experience/knowledge required to optimally price their listing
- For two-sided marketplaces such as Airbnb, optimal pricing is very important to ensure both the guests and hosts are satisfied and continue to use the platform

Solution

- Use supervised machine learning models to predict the price of a listing using the different attributes of a listing (property type, bedrooms, location, etc.)
- Airbnb can then use this as an internal pricing tool to help hosts optimally price their listing, keeping in mind the unique features of their listing

https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining#/media/File:CRISP-DM_Process_Diagram.png

Exploratory Data Analysis

Overview of the Data

Source - http://insideairbnb.com/get-the-data.html
City - New York City
Date Scrapped - March 6, 2019

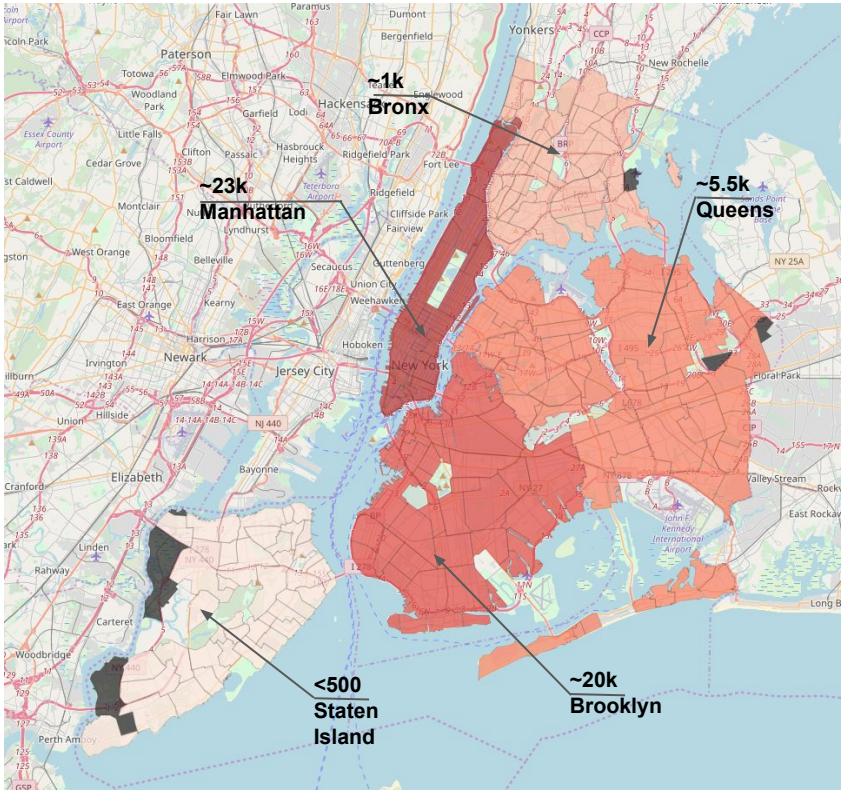
Number of Listings - 48,885
Number of Reviews - 1,115,253

Listings Table

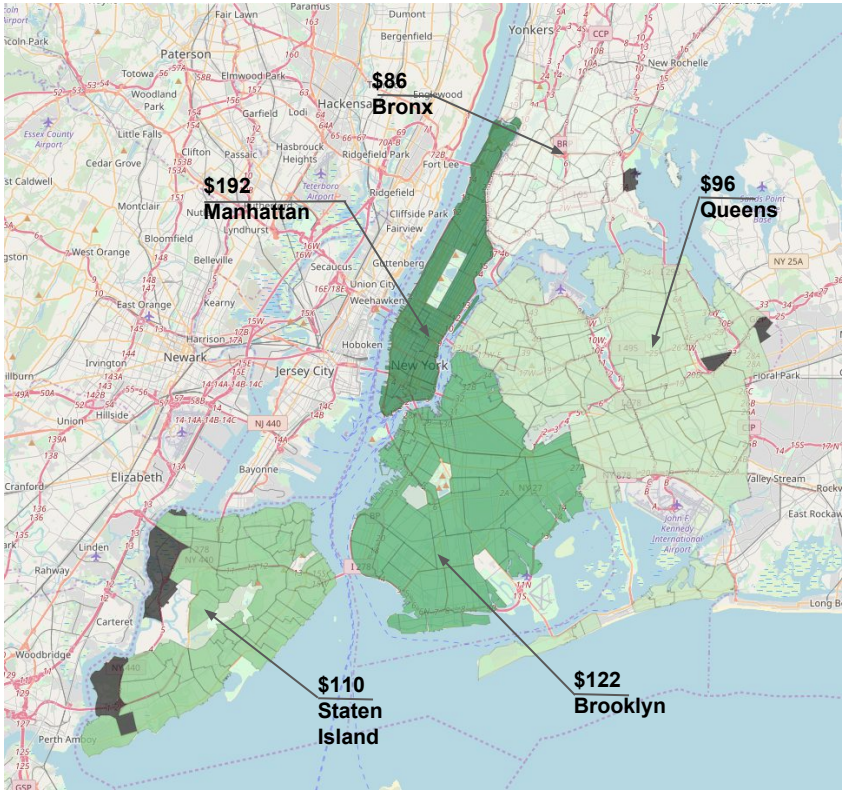
- **Listing** - ID, Name, Summary, Description, Property Type, URL, etc.
- **Host Information** - Host ID, Host Since, Host Response Rate, Host Acceptance Rate, etc.
- **Features of the Listing** - Bedrooms, Bathrooms, Max Number of People, Square Feet, etc.
- **Location Information** - City, State, Country, Zip Code, Latitude, Longitude, etc.
- **Pricing Information** - Price Per Night, Cleaning Fee, Security Deposit, Extra People Charge, etc.
- **Reviews -** Number of Reviews, First Review Date, etc.
- **Conditions -** Max Number of Nights, Minimum Number of Nights, etc.

Reviews Table

- Listing ID
- Review ID
- Date Posted
- Reviewer ID
- Reviewer Name
- Comments

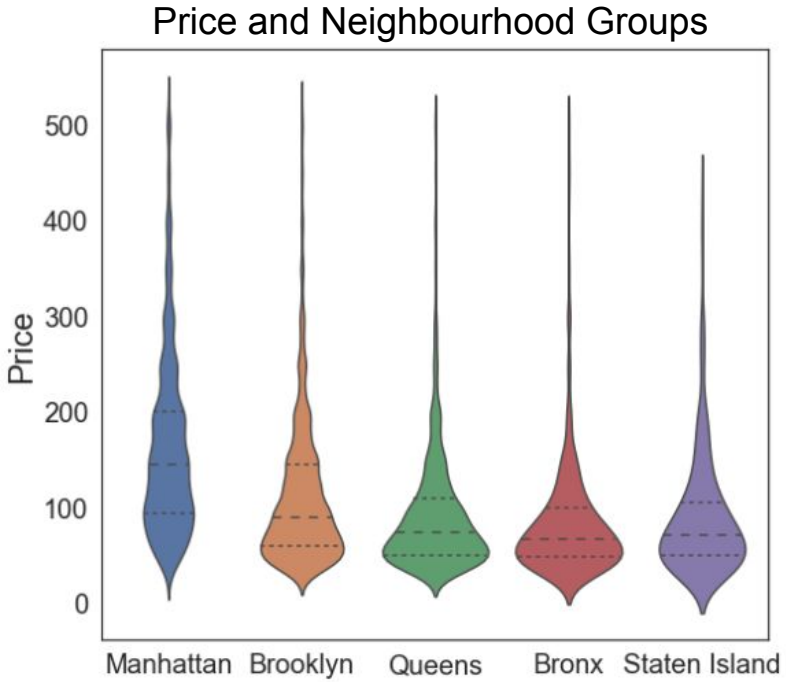


Concentration of Listings - Manhattan has the most listings, followed by Brooklyn and then Queens and surrounding neighborhoods.

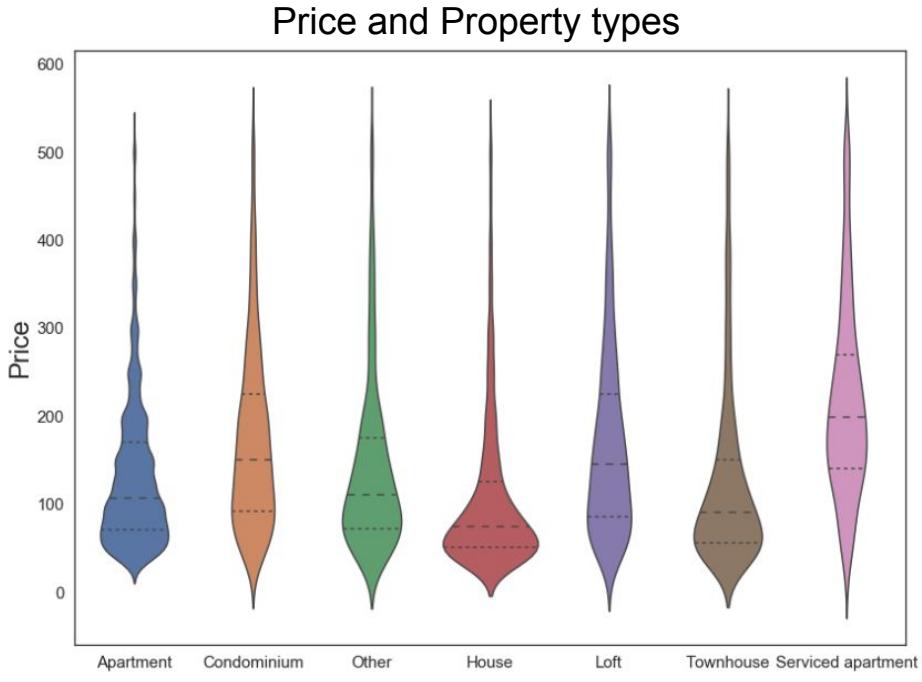


Most Expensive Neighborhoods - Manhattan has the most expensive listings, followed by Brooklyn and Staten Island

Neighbourhood Group and Property Type

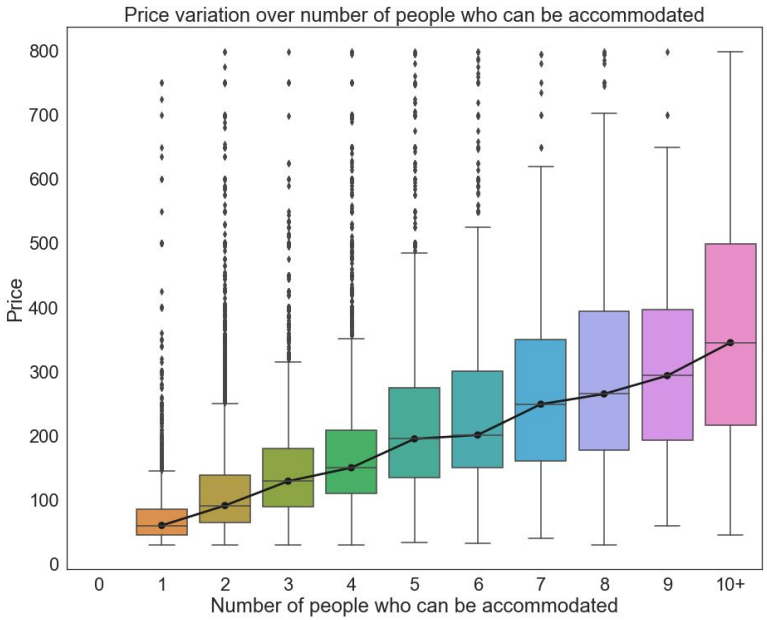


Manhattan and Brooklyn tend to have higher prices than other neighborhoods.

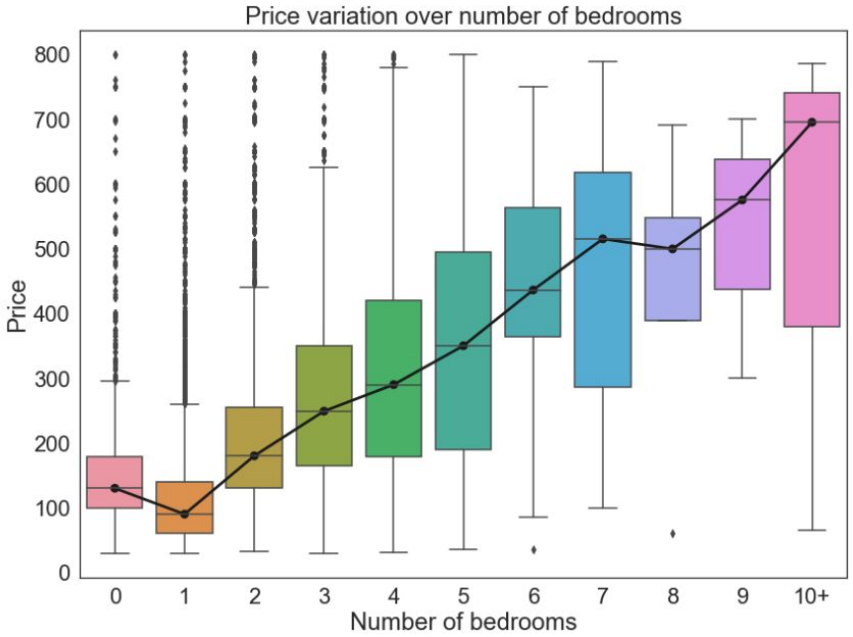


Serviced Apartments and other types of apartments tend to have higher prices than Houses or Townhouses.

Size of Accomodation

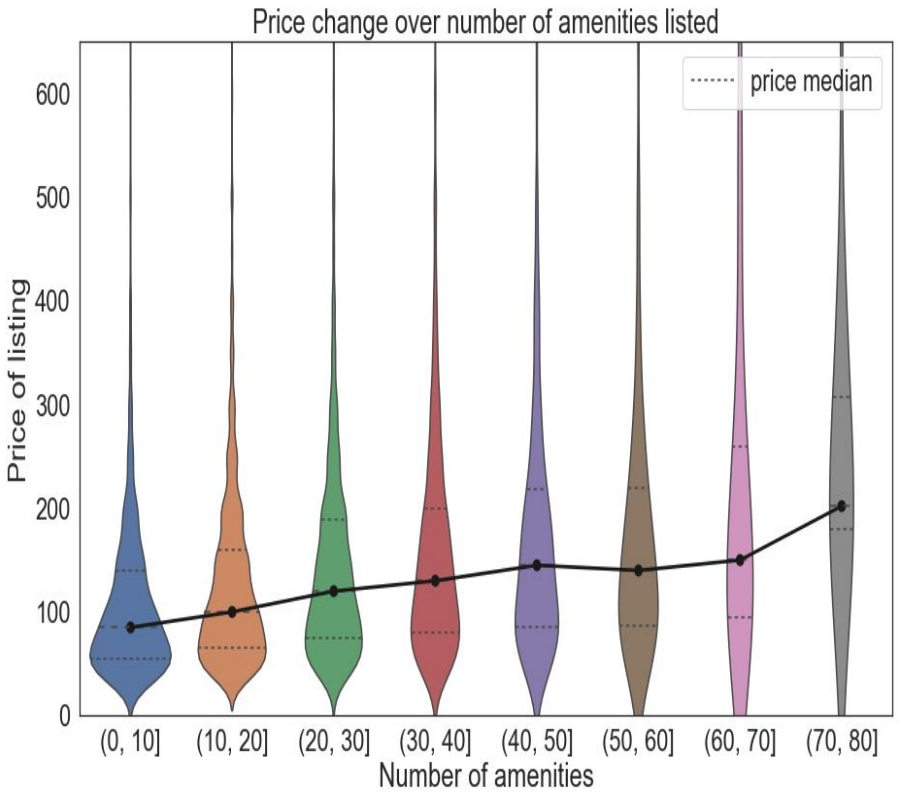


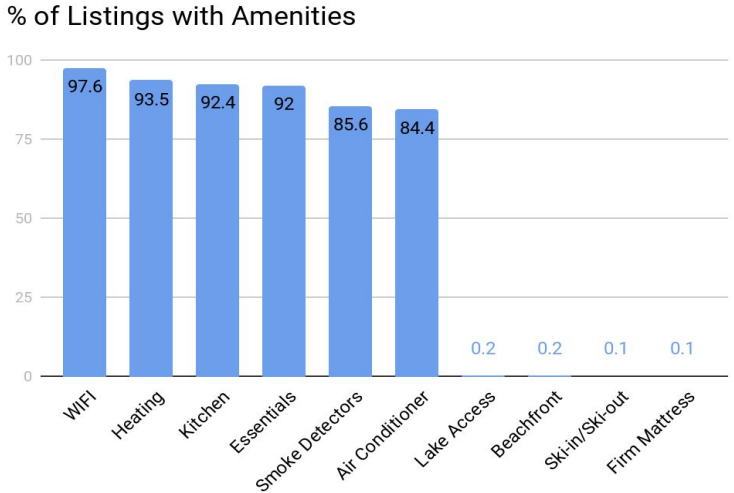
As expected, prices tend to increase as the listing maximum accomodation size increases.

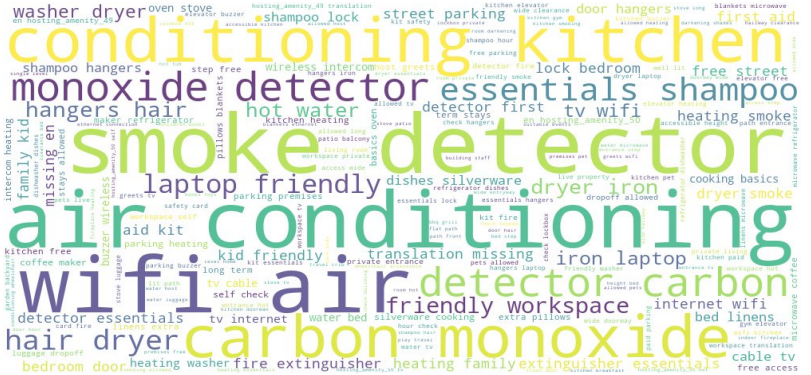


Prices also tend to increase as the number of bedrooms in a listing increase.

Amenities Overview

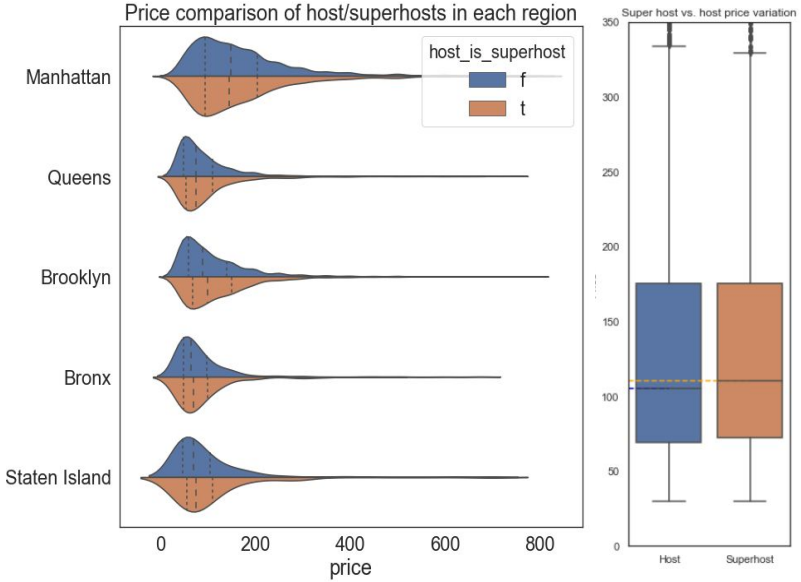


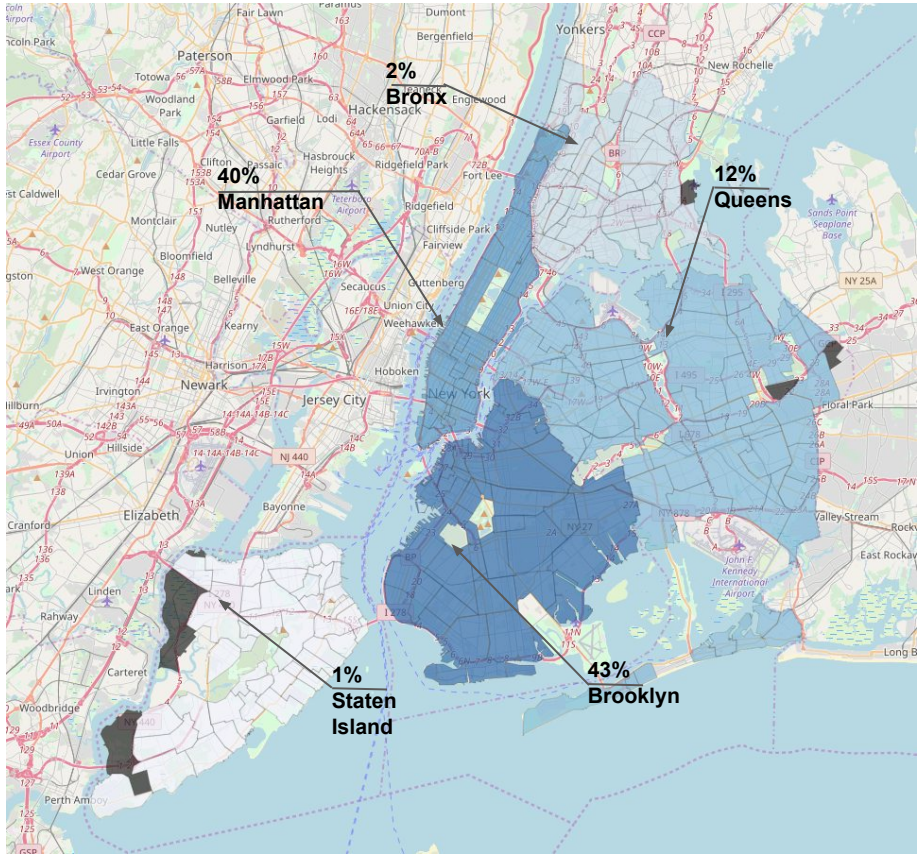




Concentration of Superhosts

Superhosts are experienced hosts who provide a shining example for other hosts, and extraordinary experiences for their guests. They maintain a response rate of 90% or higher and maintain an overall rating of at least 4.8/5.0.

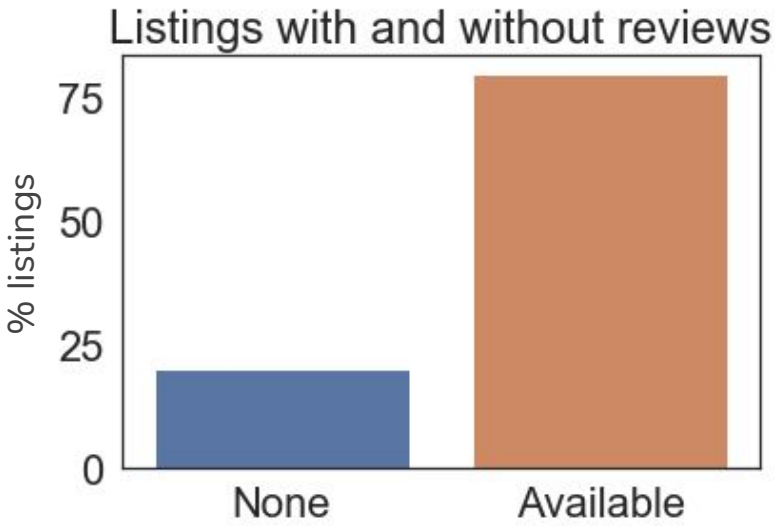


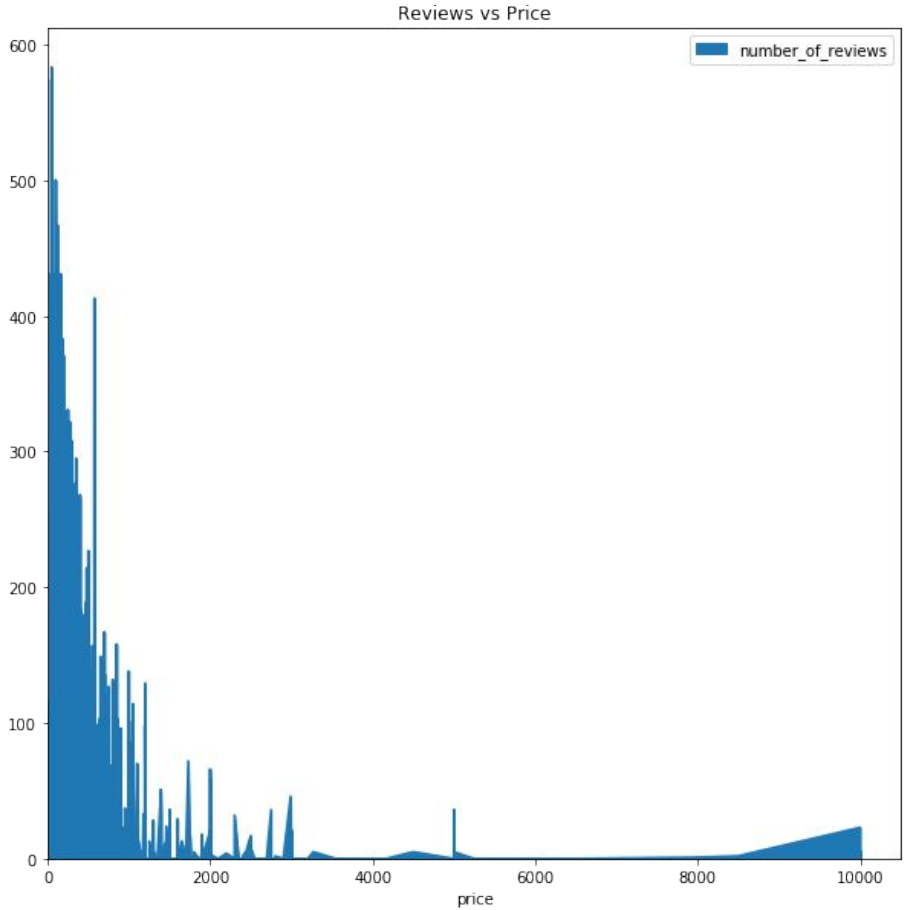


Concentration of Superhosts across NYC

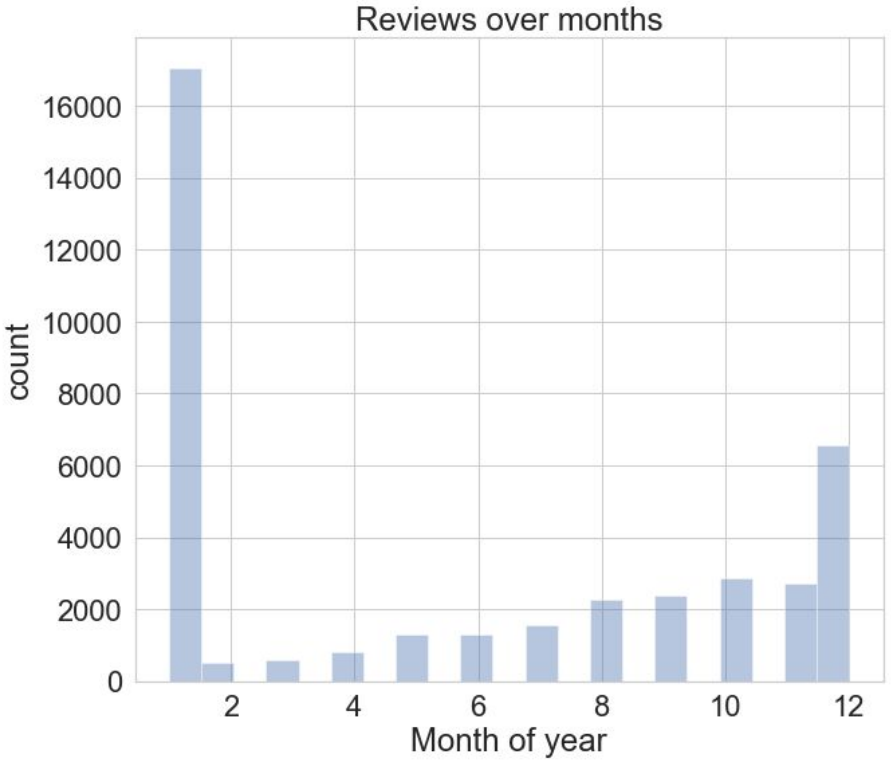
Reviews and Listings

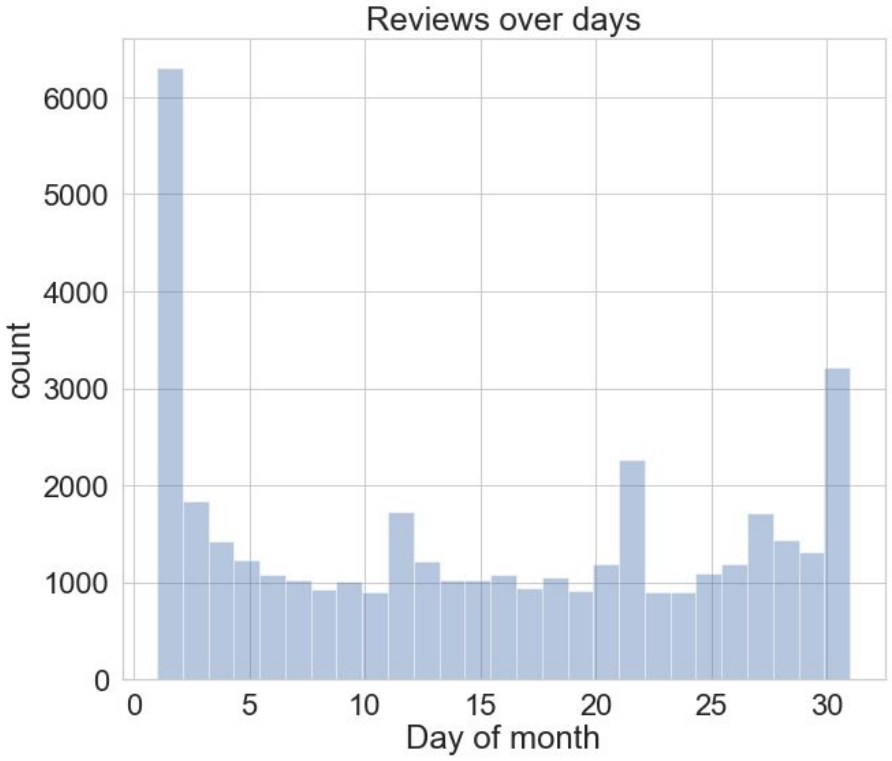
- Lower priced listings tend to get more activity and feedback than higher priced ones
- Nearly 25% of the listings do not have even a single review



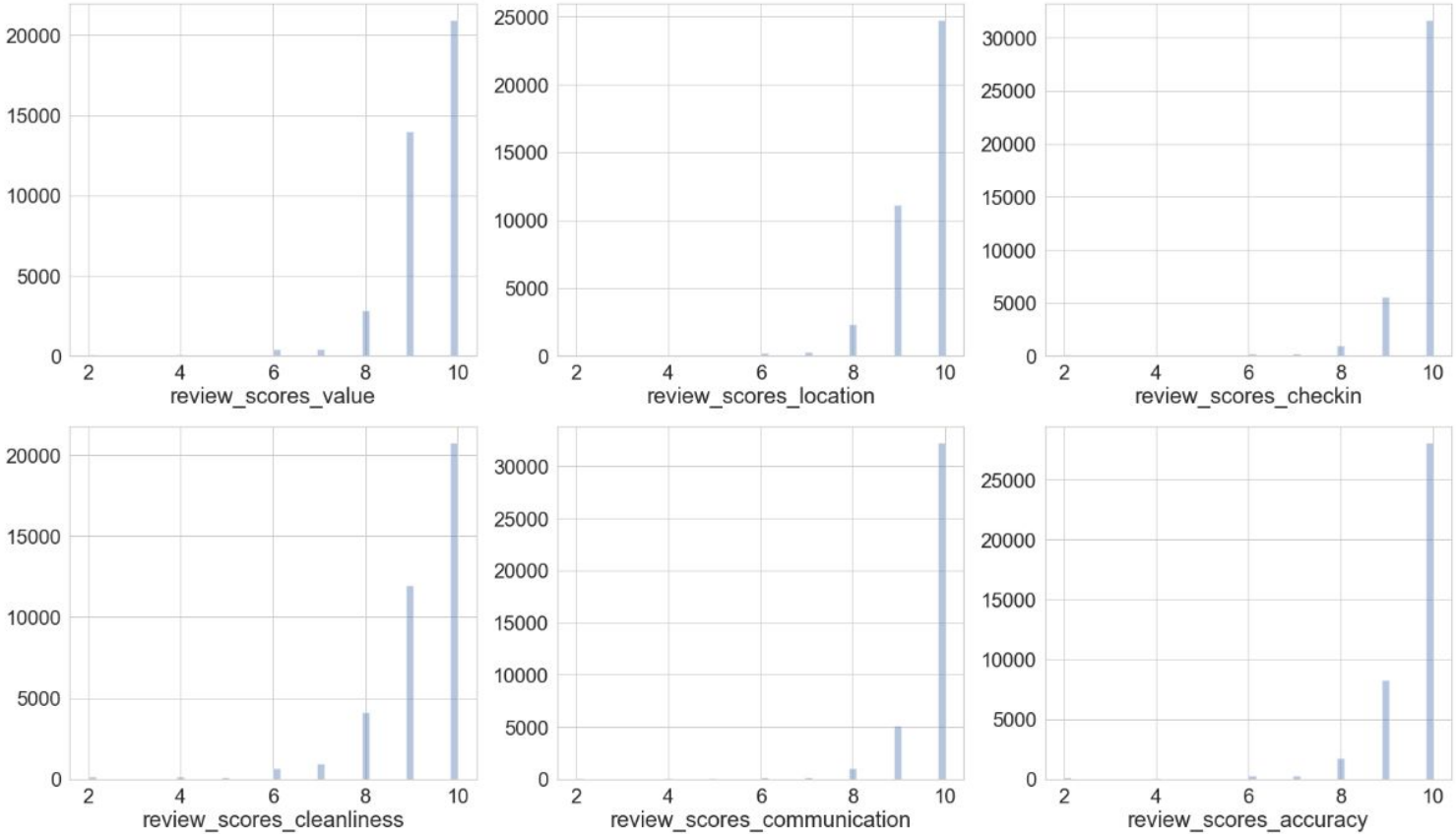


Reviews Frequency in a Year





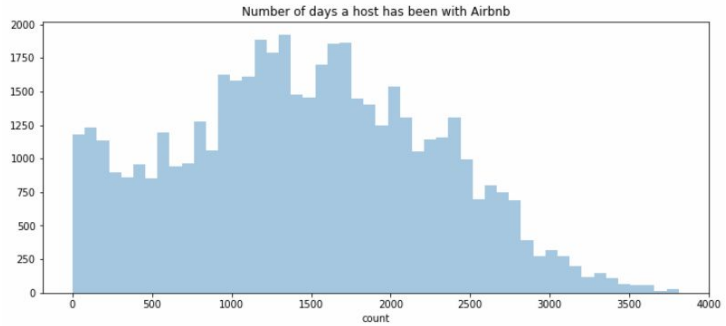
Reviews Distribution

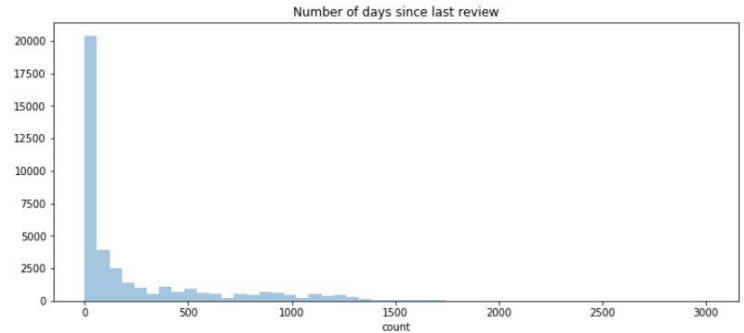


Feature Engineering

We tried deriving variables using existing features to augment the dataset. To this end, two additional features were computed using existing features.

- ***ndays_host*** : This column contains the number of days a host has been with Airbnb
- ***ndays_last_review*:** This column contains the number of days that have passed since a review was posted for the listing.





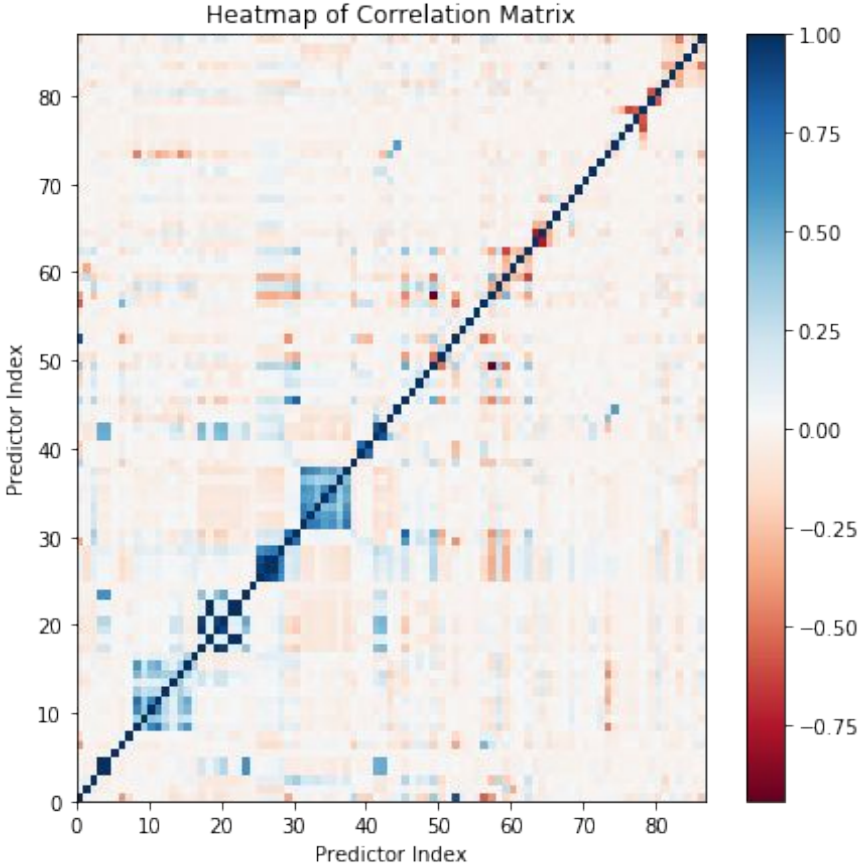
Correlated Feature Selection

Features that were removed due to multicollinearity:

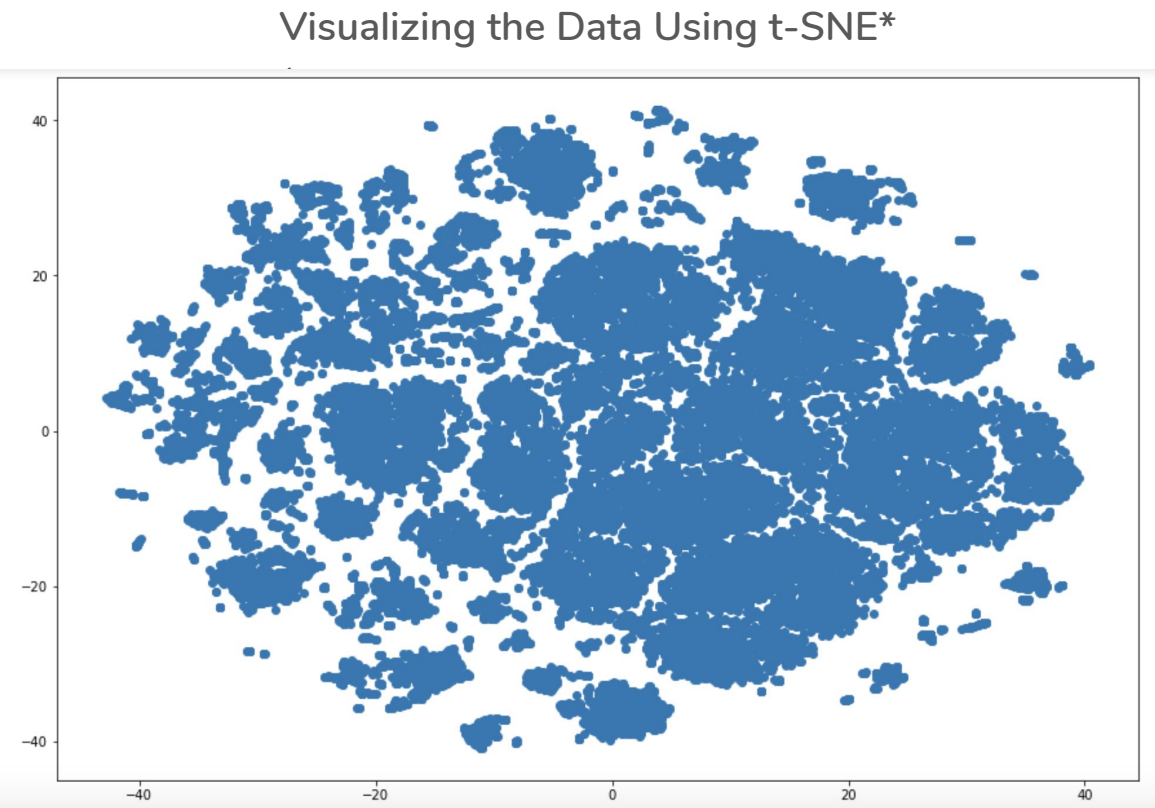
- availability_60
  availability_90
  availability_365

- host_total_listings_count
  calculated_host_listings_count_entire_homes
  calculated_host_listings_count_private_rooms
  calculated_host_listings_count_shared_rooms

- minimum_minimum_nights
  minimum_maximum_nights
  maximum_maximum_nights
  maximum_nights_avg_ntm



Unsupervised Learning - t-SNE

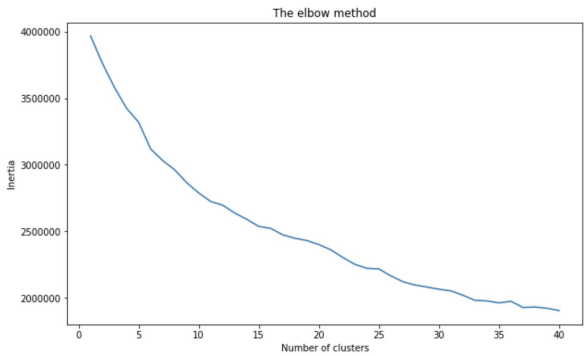


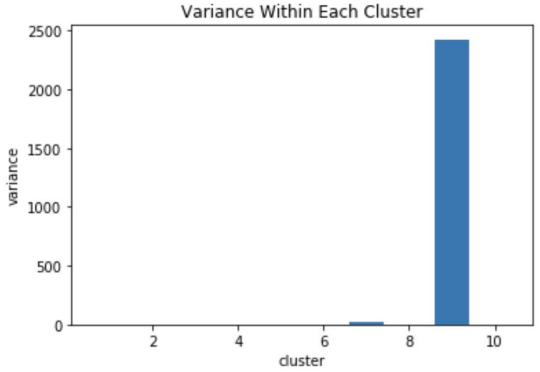
* *learning_rate = 100*

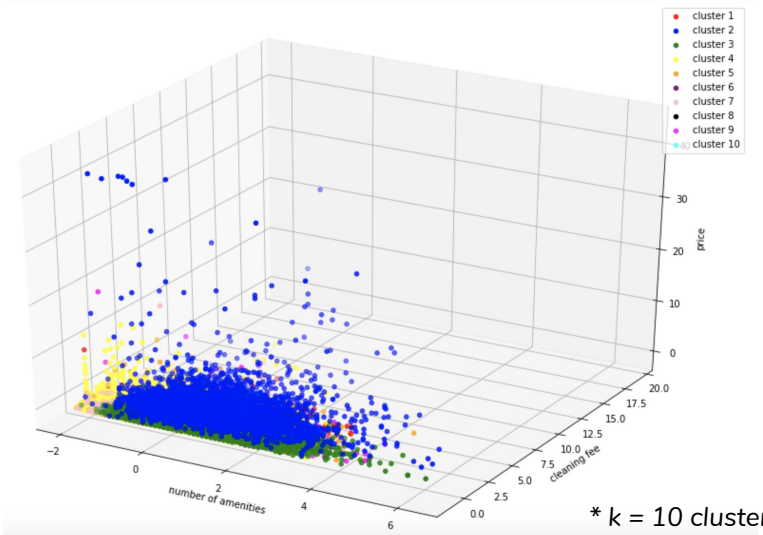
After tuning the learning rate and running several different iterations, no overarching clusters were seen in the data.

t-SNE helped us quickly visualize our data in two dimensions to see if there were any specific clusters we should look at before we started any supervised learning.

Unsupervised Learning - K-Means Clustering







* *k = 10 clusters*

K-Means clustering helped further confirms the result from t-SNE that there are no specific clusters in the data that we should consider before proceeding to supervised learning methods.

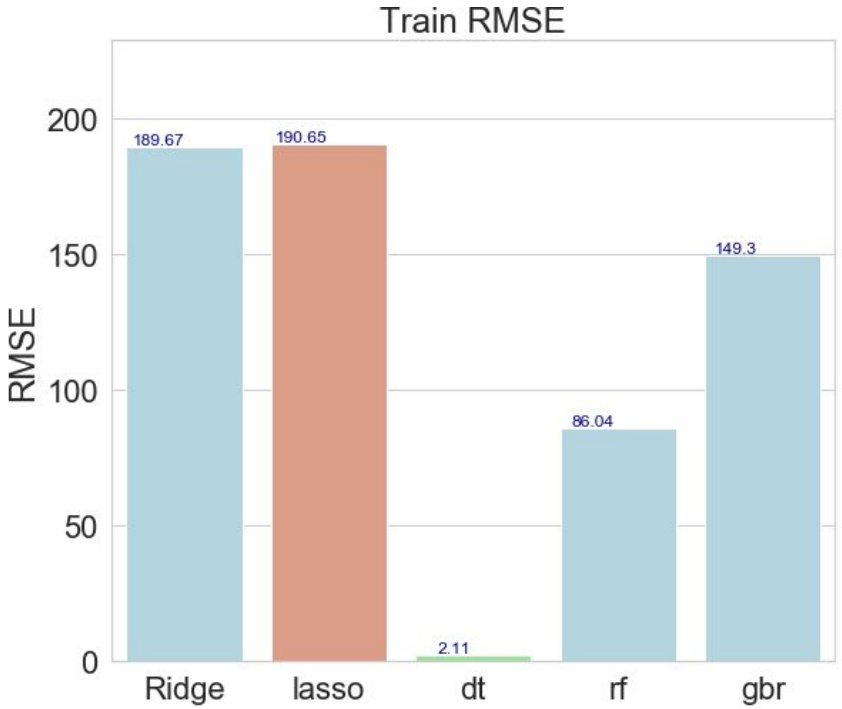
Conclusion from Unsupervised Exploration - We will perform supervised learning on the entire dataset as no specific clusters were identified in the unsupervised learning phase.

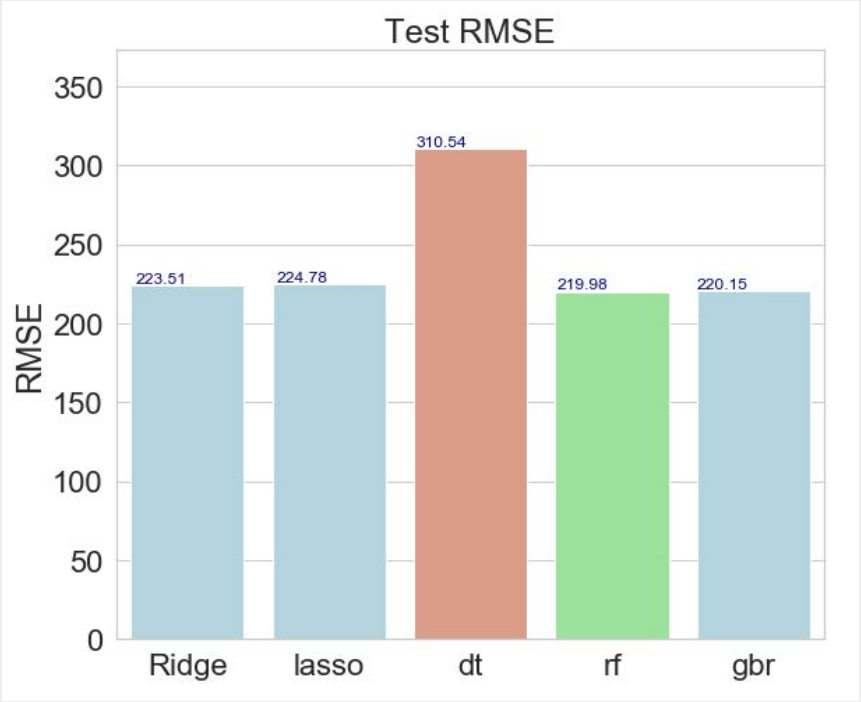
Initial ML Models

Machine Learning Algorithms Used

- **Ridge Regression**
  - Technique for analyzing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors and gives more accurate results.
- **Lasso Regression**
  - Technique to perform both regularization and feature selection. The only difference from Ridge Regression is instead of taking the square of the coefficients, magnitudes are taken into account. This type of regularization (L1) can lead to zero coefficients i.e. some of the features are completely neglected for the evaluation of output. So Lasso regression not only helps in reducing overfitting, but it can also help in feature selection.
- **Decision Trees**
  - Non-parametric supervised learning method used for classification and regression. Decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules.
- **Random Forests**
  - Ensemble learning method for classification or regression that operates by constructing a multitude of decision trees at the time of training and outputting the mode of the classes (in classification) or mean prediction (in regression) of the individual tree. Random forests can correct for decision trees' habit of overfitting the training set.
- **Gradient Boosting**
  - Ensemble method for classification or regression, which produces a prediction model in the form of an ensemble of weak prediction decision trees. It builds the model in a stage-wise fashion and generalizes the model by allowing for optimization of a specific loss function, which allows weak learners to become better learners.

Initial Results - RMSEs



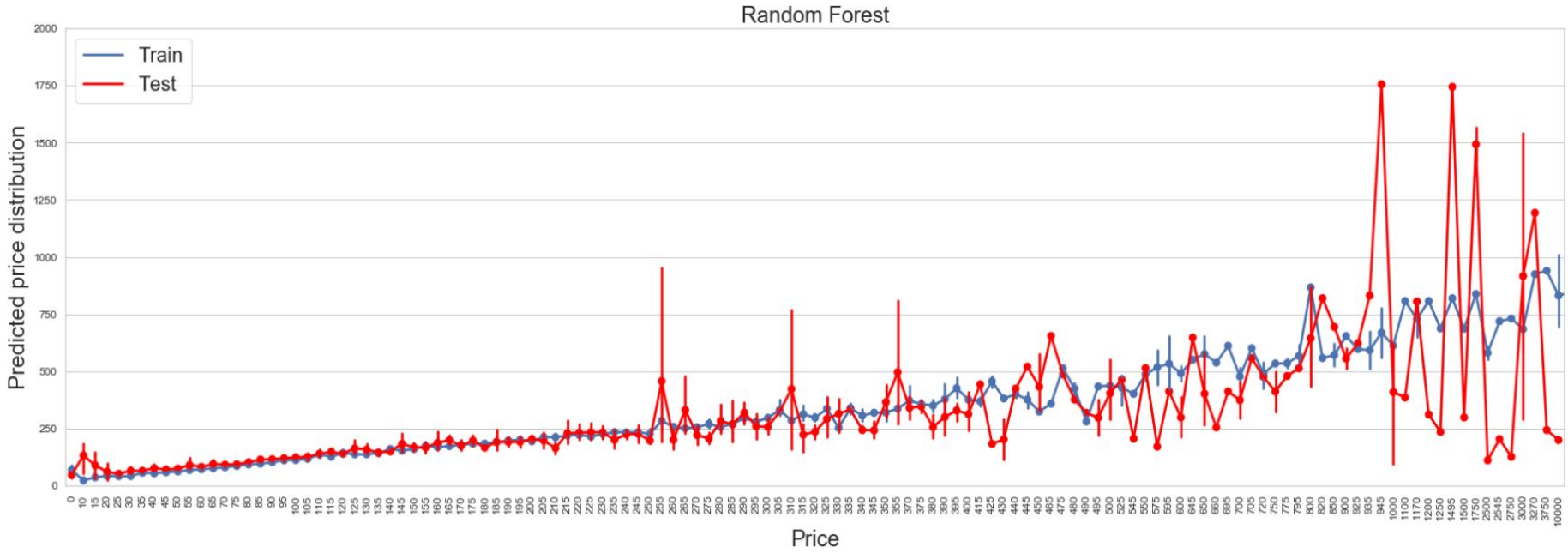


Initial Results - Random Forest

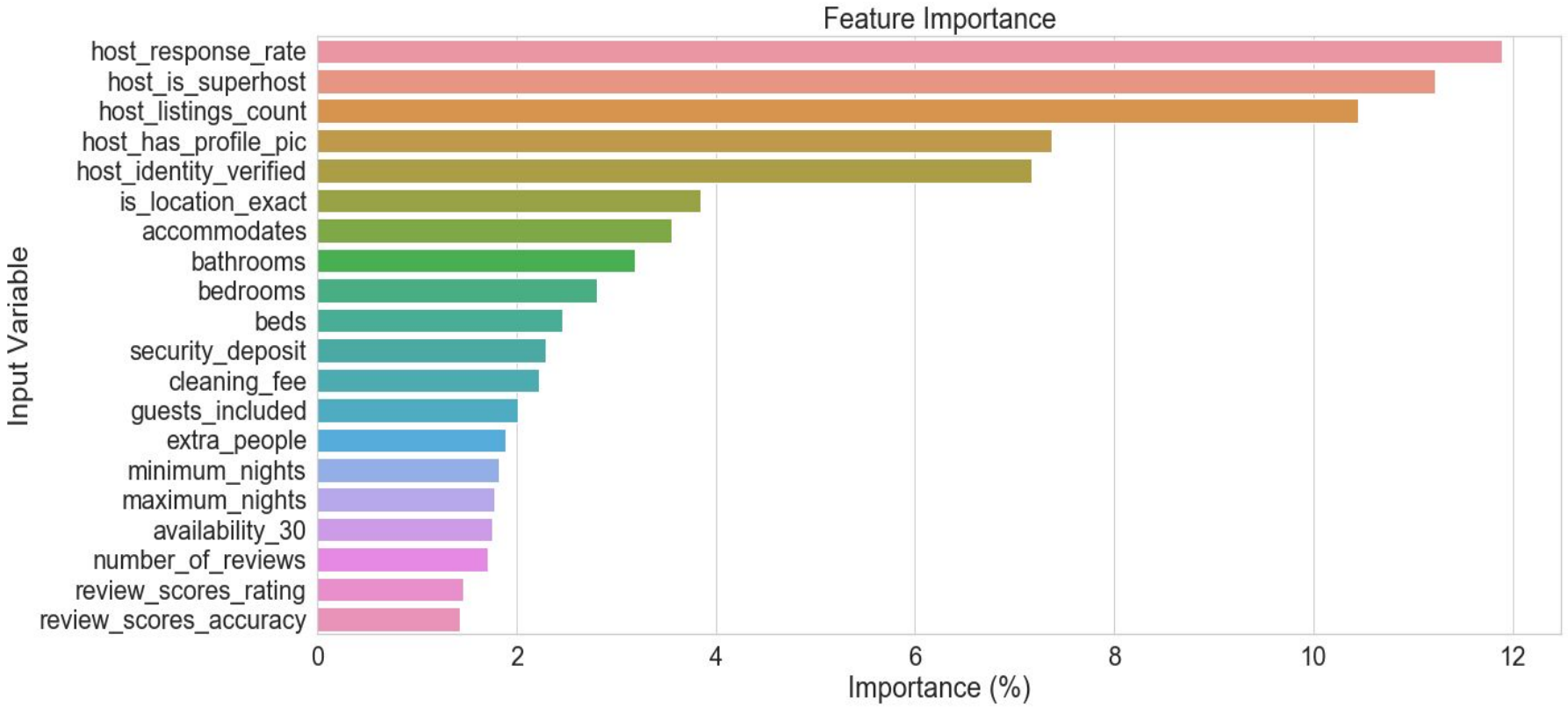
A Random Forest Regressor was run on the cleaned data with default parameters

Train Set RMSE : 86.04

Test Set RMSE: 219.98

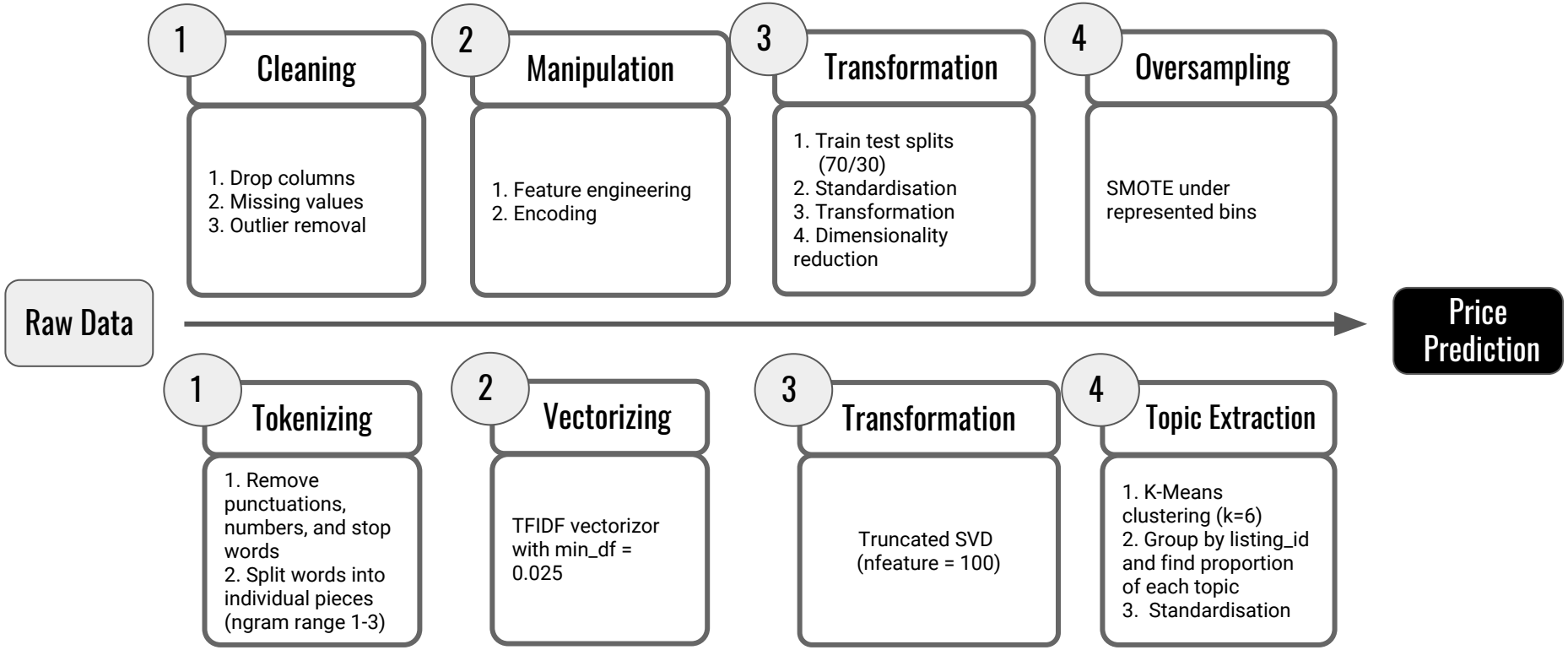


Feature Importance - Random Forest

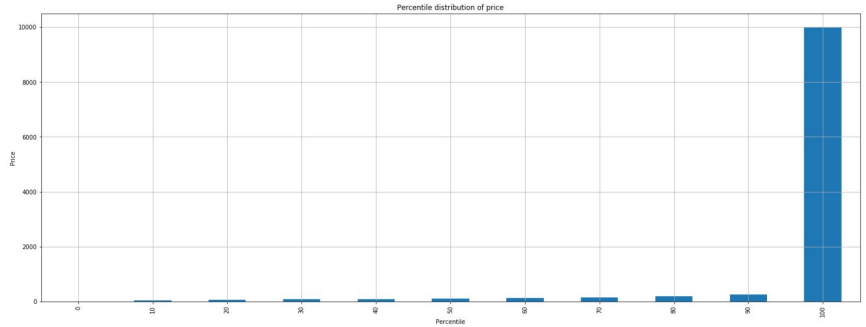


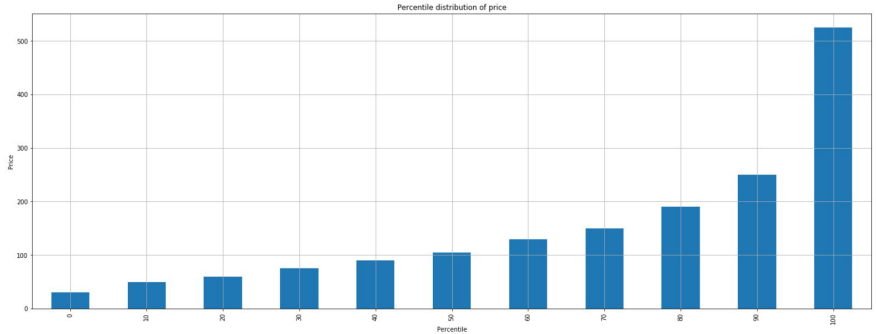
Data Manipulation

Pipeline


Raw Data
1
Cleaning
1. Drop columns
2. Missing values
3. Outlier removal
2
Manipulation
1. Feature engineering
2. Encoding
3
Transformation
1. Train test splits (70/30)
2. Standardisation
3. Transformation
4. Dimensionality reduction
4
Oversampling
SMOTE under represented bins
Price Prediction
1
Tokenizing
1. Remove punctuations, numbers, and stop words
2. Split words into individual pieces (ngram range 1-3)
2
Vectorizing
TFIDF vectorizor with min_df = 0.025
3
Transformation
Truncated SVD (nfeature = 100)
4
Topic Extraction
1. K-Means clustering (k=6)
2. Group by listing_id and find proportion of each topic
3. Standardisation

Percentile Distribution of Price

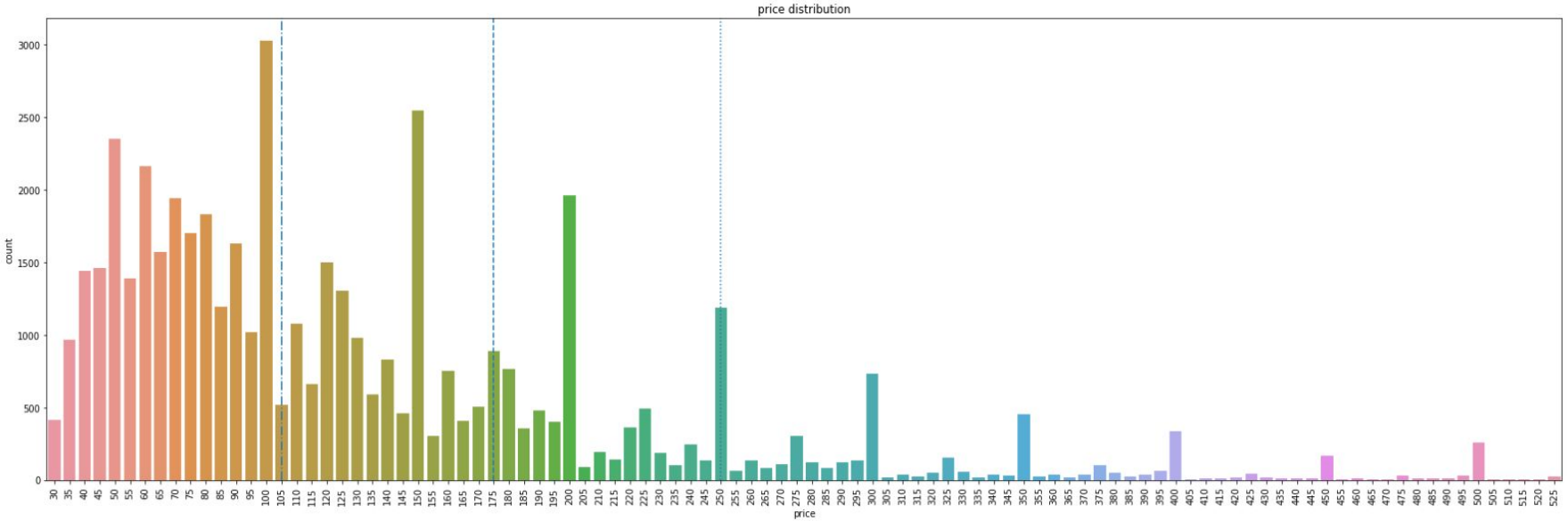




The top 2 percentile of prices had an enormous skew, and constituted around 2% of the overall dataset alone. These prices were larger by around 2 orders of magnitudes and were considered as outliers.

Upon filtering out outliers, the percentile distribution of prices was more consistent.

Distribution of Price



Majority of the listings are below the 75% percentile of the price distribution

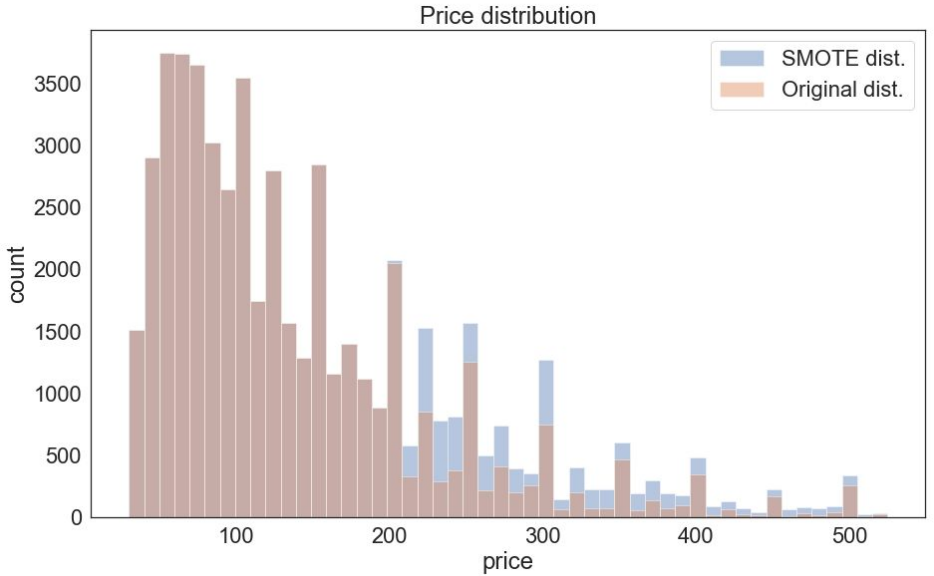
Periodicity at every step of $50

Oversampling Under-represented Price Bins

Nearly 85% of all listings in the dataset were less than or equal to 200$, with only 15% spread out from $205 to $525 .

To compensate for this skew, we oversampled the latter price bins using SMOTE.

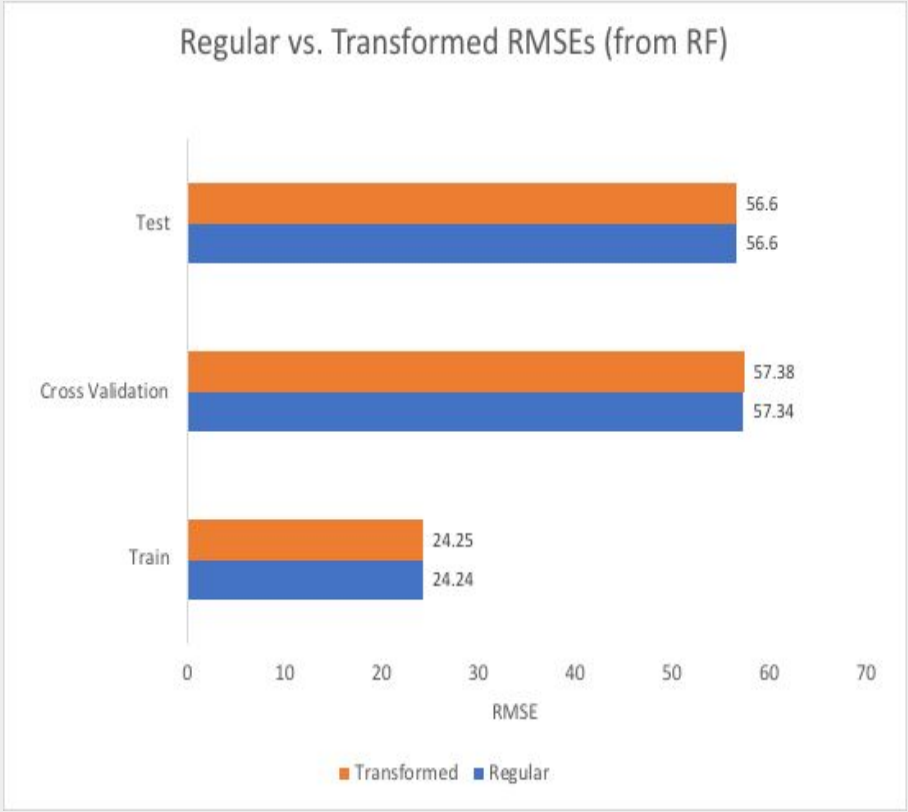
Synthetic Minority Over-sampling Technique (SMOTE) was used to synthetically generate additional samples for the latter price bins to get a more reasonable distribution



Transformations

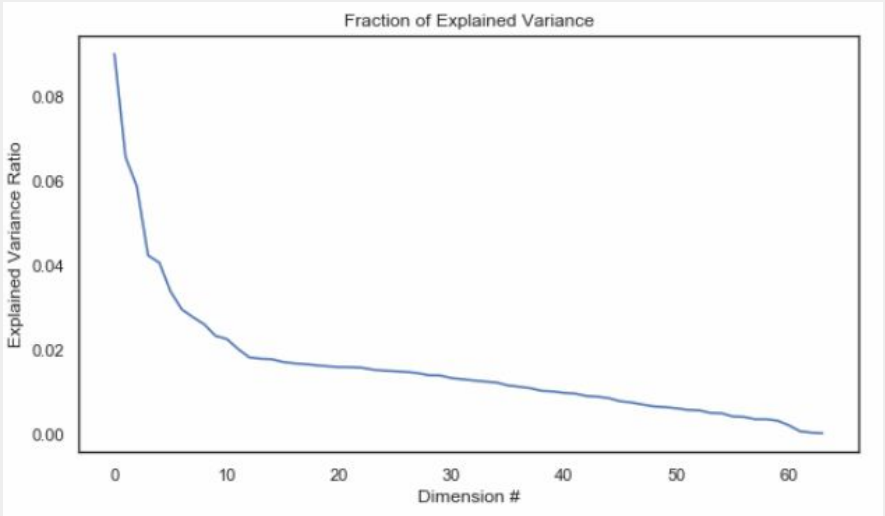
- We tried to do some more feature engineering by creating different transformations of the predictor variables. Each predictor variable had 4 versions:
  - Original variable
  - Log Transformed
  - Square Transformed
  - Square Root Transformed

- Transformations did not have any significant positive impact on the RMSEs when compared to the regular model. Being mindful of not accentuating the curse of dimensionality, we decided not to proceed with the transformations.



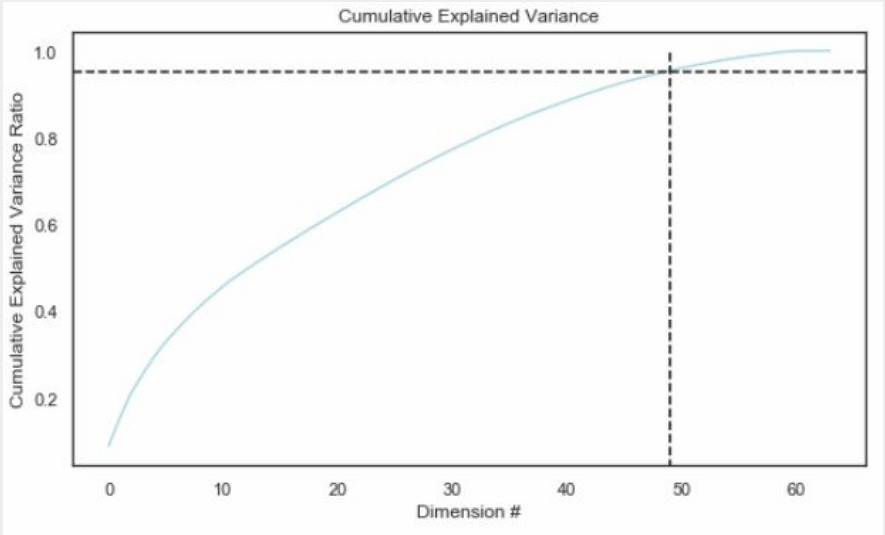
Dimensionality Reduction - PCA

Dimensions vs. Explained Variance



- Plotting the explained variance ratio against no. of dimensions gives a nice Elbow curve

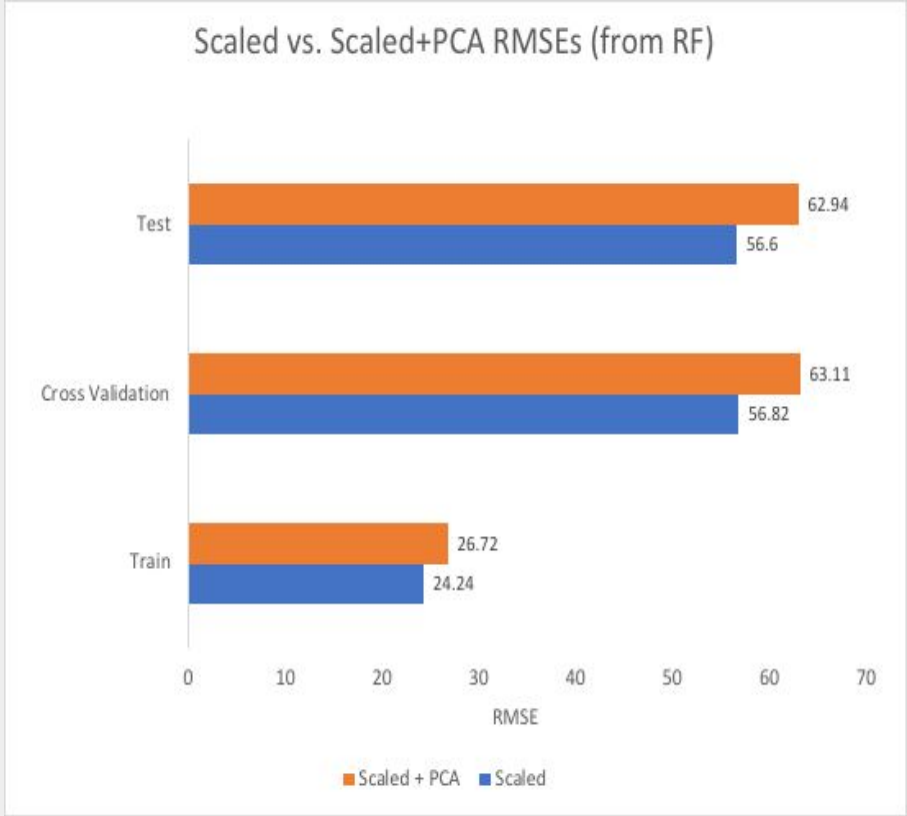
Dimensions vs. Cumulative Variance



- Number of principal components covering 95% of variance : 49

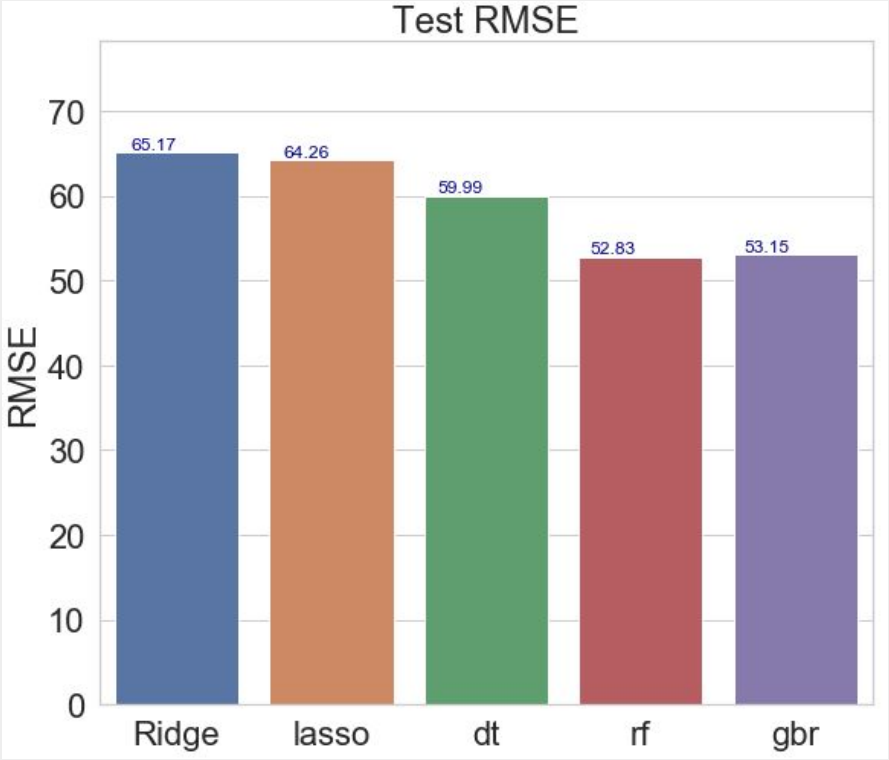
PCA Results

- We tried to do some dimensionality reduction on our dataset by running principal component analysis (PCA)
- PCA did not have a positive impact on the RMSEs when compared to the regular scaled model, in fact it performed worse
- We were bound to lose some predictive power as we kept only 49 variables that explained 95% of total variance
- However, since there was no significant reduction in number of variables, we decided not to proceed with the components



Final ML Models

Algorithm Comparison

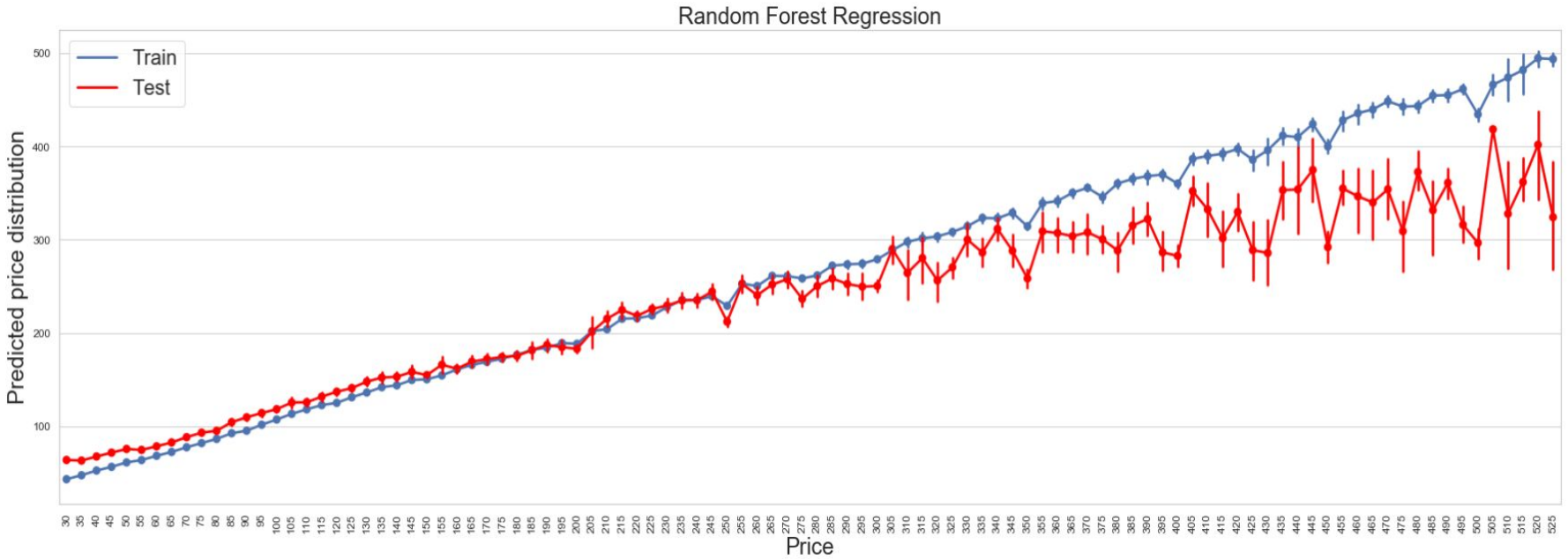


Random Forest Regression

A Random Forest regressor was run on the cleaned data with default parameters

Train Set RMSE : 22.4

Test Set RMSE 52.83

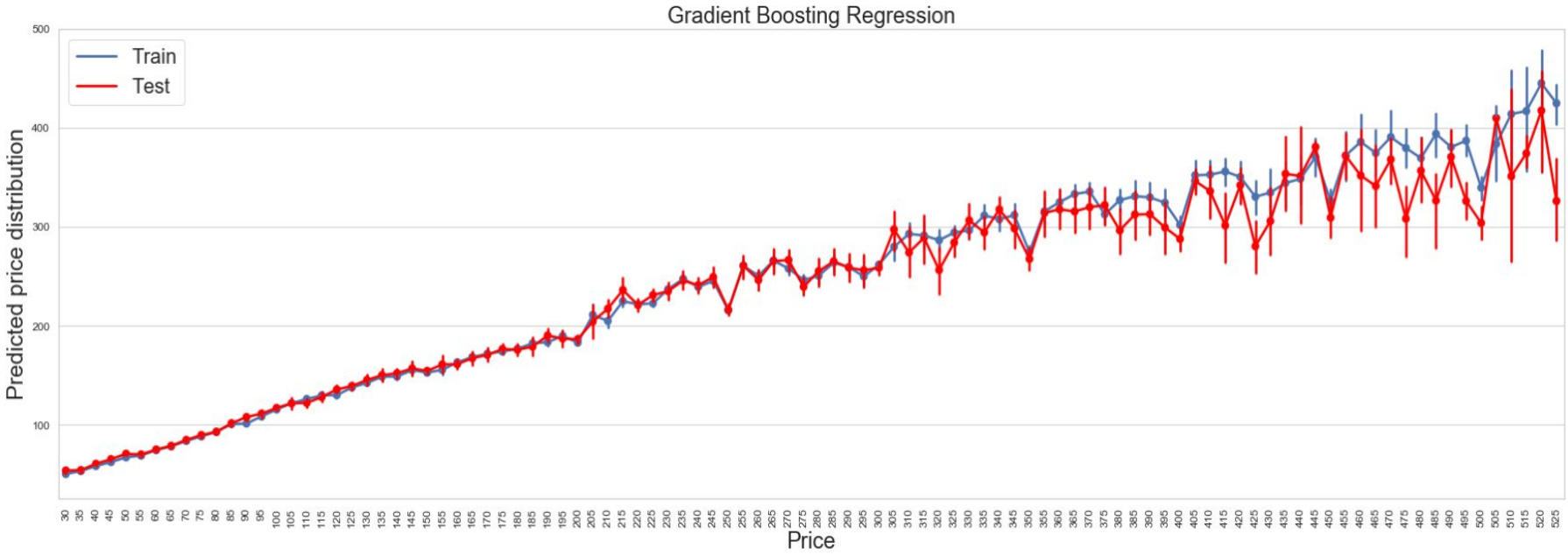


Gradient Boosting Regression

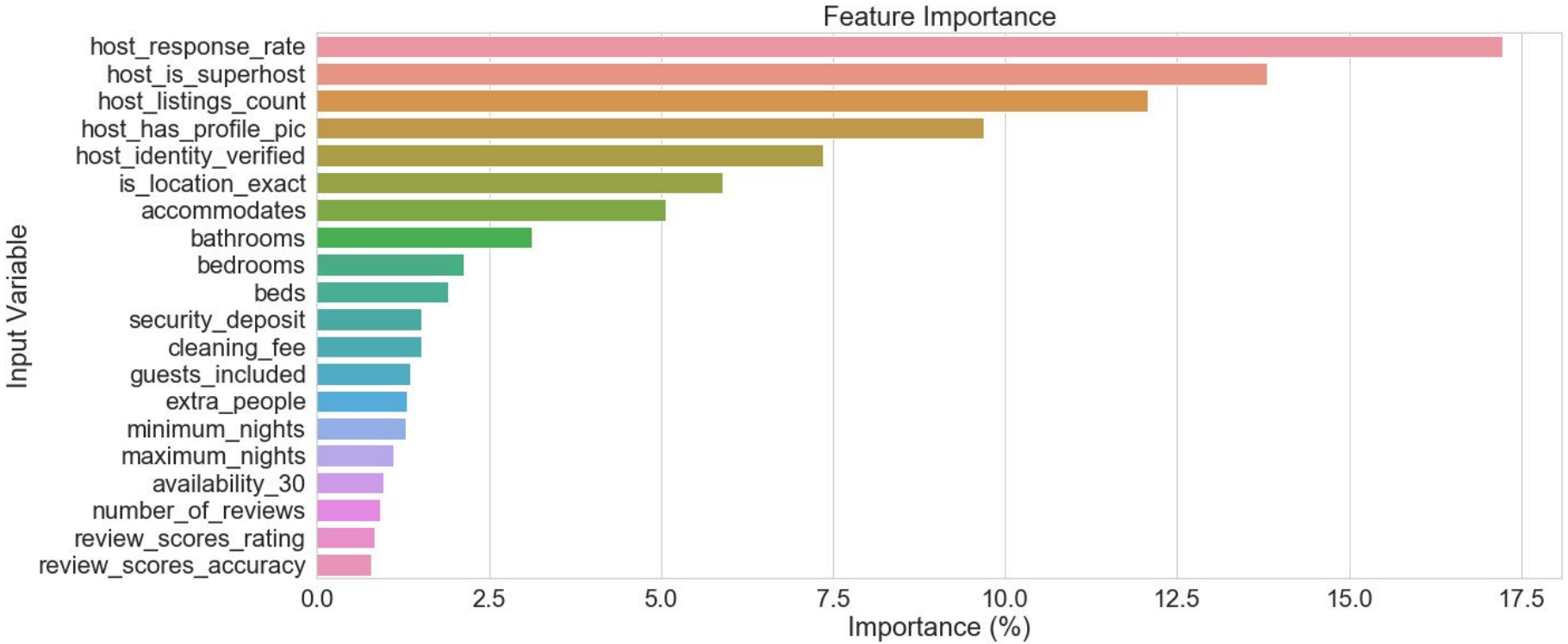
A Gradient Boosting regressor was run on the cleaned data with default parameters

Train Set RMSE : 46.3

Test Set RMSE 53.15



Feature Importance - Gradient Boosting



Hyperparameter Tuning

Ridge Regression

Hyperparameters:
alpha = 1e-08

Train Set RMSE: 64.40
Test Set RMSE: 65.17
Cross Validation RMSE: 64.53

Lasso Regression

Hyperparameters:
alpha = 0.15

Train Set RMSE: 64.46
Test Set RMSE: 64.26
Cross Validation RMSE: 64.58

Decision Trees

Hyperparameters:
max_depth = 14
max_features = 50
min_samples_split = 0.01
min_samples_leaf = 0.001

Train Set RMSE: 58.78
Test Set RMSE: 59.99
Cross Validation RMSE: 61.04

Random Forest

Hyperparameters:
bootstrap=False
criterion='mse'
max_depth=40
max_features='sqrt'
min_samples_split=5

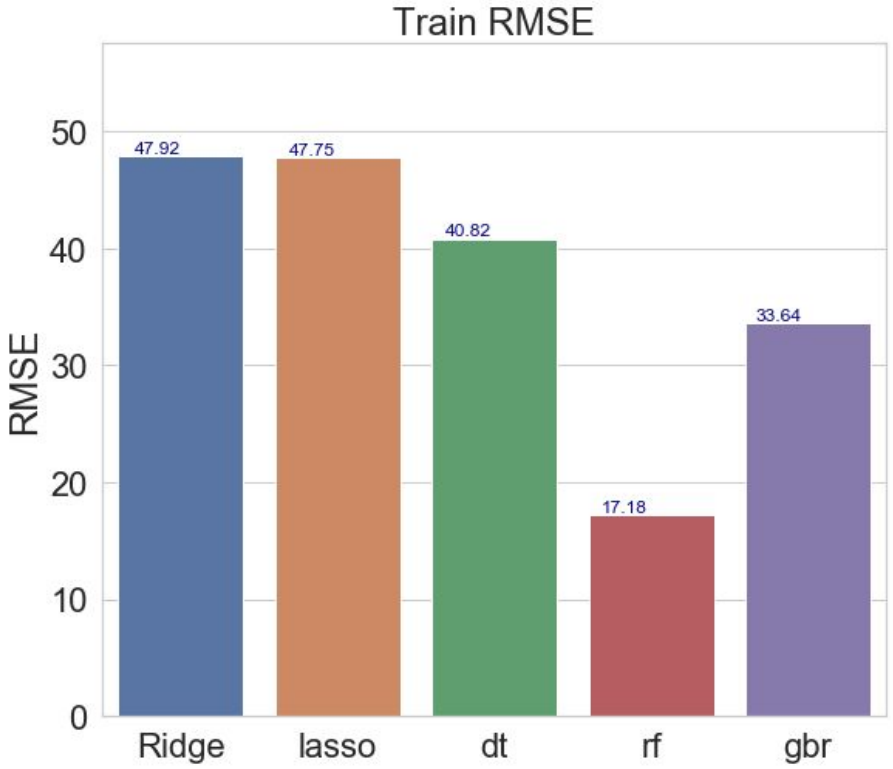
Train Set RMSE: 10.60
Test Set RMSE: 52.52
Cross Validation RMSE: 52.88

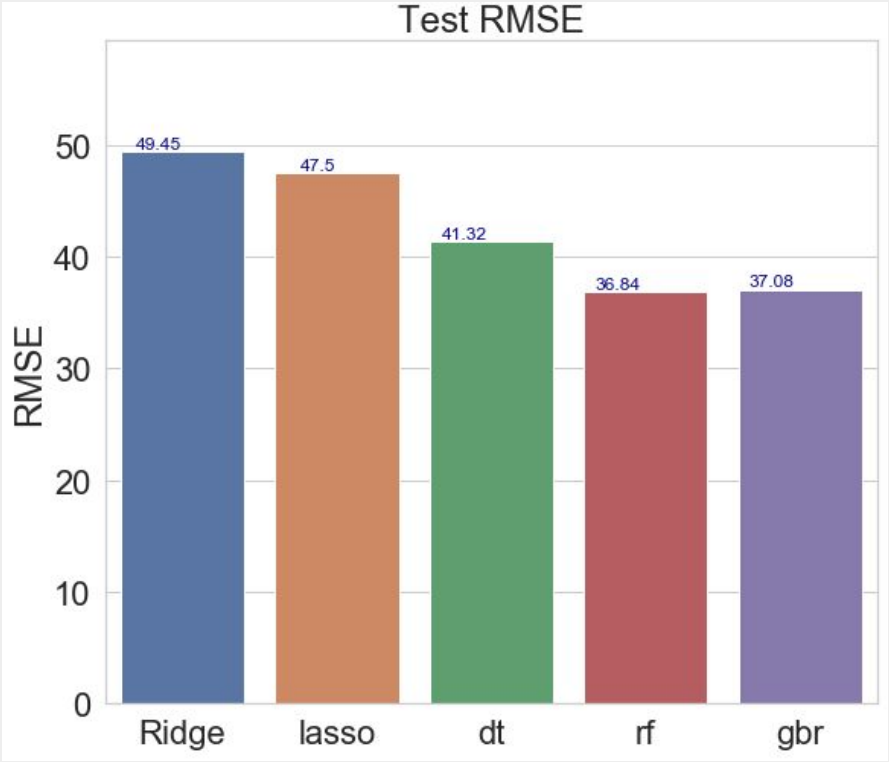
Gradient Boosting

Hyperparameters:
subsample = 1,
max_depth=5,
max_features="sqrt",
min_samples_split=35,
learning_rate=0.1

Train Set RMSE: 46.3
Test Set RMSE: 53.15
Cross Validation RMSE: 52.94

Algo Comparison - 80% of Data





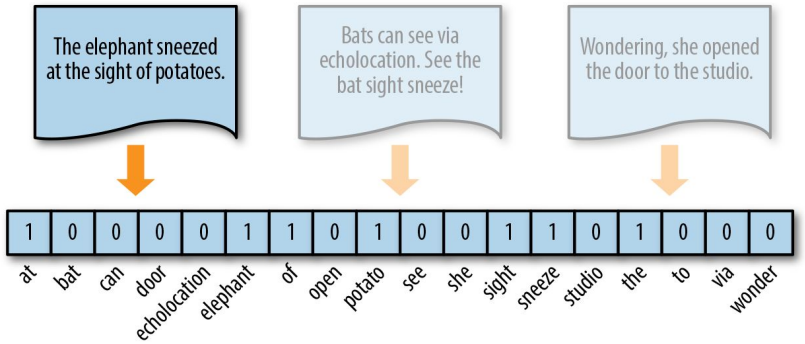
NLP Analysis

Dealing with Text Data

Need to turn entries in a text column into vectors of the same length, so that models can use them.

Popular methods include count, tf-idf, word2vec/doc2vec, etc.

We used tf-idf combined with truncated SVD to vectorize text and have manageable amount of columns.



Text Vectorizing - Beware of memory usage

Text Preprocessing

Sklearn's text vectorizers have built-in features to clean text such as stopword removal and case-lowering. Also, can customize tokenizer using regular expression

With over 1 million reviews, need to be conscious about memory usage

TfidfVectorizer

Set parameter df (document frequency) to higher than 0 to conserve memory and filter out noise, but not so high so that important information gets lost

Use a larger ngram range can help gain information that's encoded in multiple words, but these ngrams tend to have low document frequency so need to calibrate

TruncatedSVD

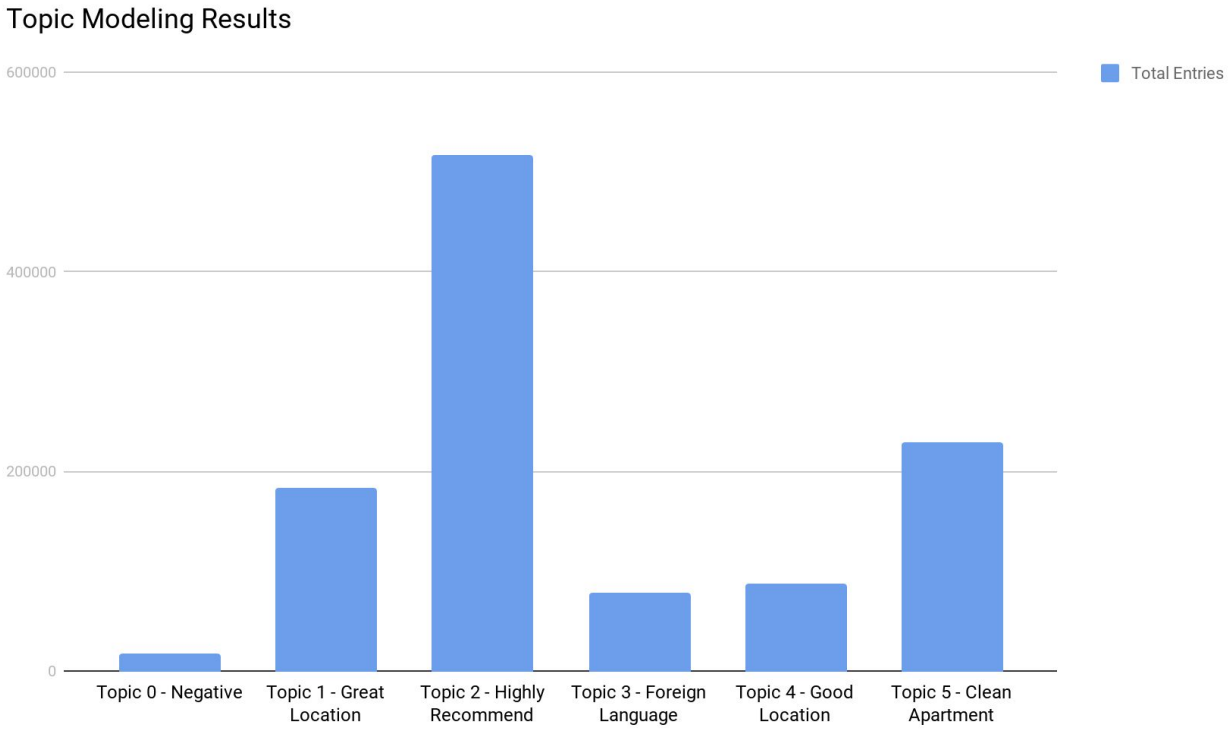
Reduce number of columns returned by TfidfVectorizer

Can use factor loadings to interpret how each word matters in the column

Simple Topic Clustering with K-Means

- This happens after we assign a vector to each review entry after text processing steps outlined in the ‘previews’ slide
- Used K-Means to cluster topics with K=6

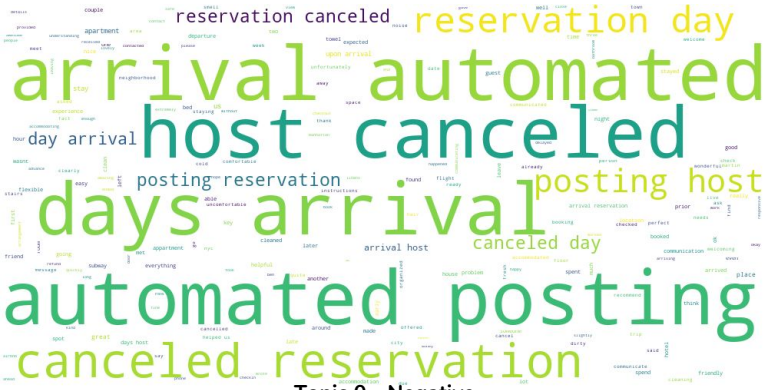
K-Means Topic Modeling Results



- Reviews are overwhelmingly positive with a very small number of negative reviews (1.6% of total)
- Foreign language constitutes another meaningful cluster (7.1% of total)

K-means topic clustering does a good job with separating the different themes in the treviews

- While not the most sophisticated method, k-means does a relatively good job of differentiating high level themes such as negative reviews and foreign language reviews
- However, it does a less good job of differentiating between positive reviews and the differences are less interpretable
- This method with TFIDF only looks at word or n-gram frequency thus can have more trouble deciphering latent meanings compared with something like word2vec/doc2vec



Topic 0 - Negative

Topic 3 - Foreign language

Convert all reviews of a listing to a vector for modeling

For each listing, aggregate and find proportion of each label

This would generate a vector of length 6 for each listing, and can serve as predictors for the final model

Can consider combining some of the similar topics together as they are collinear

	kmeans_label	listing_id	id	comments
0	2	4218034	113949495	We loved our stay at the Urban Jungle! The dup...
1	4	20817201	314215888	Is a good place just to sleep and go. Perfect ...
2	2	4152752	185785747	Niko's place is in a centralized area near all...
3	1	20990607	211627258	Angie is a great host, accommodating, very swe...
4	2	1759462	66002562	The place is lovely, and the location is amazi...

listing_id	topic_0	topic_1	topic_2	topic_3	topic_4	topic_5
2454	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2539	0.000000	0.222222	0.777778	0.000000	0.000000	0.000000
2595	0.000000	0.232558	0.302326	0.093023	0.046512	0.325581
3330	0.051282	0.102564	0.435897	0.051282	0.051282	0.307692
3831	0.004329	0.134199	0.571429	0.121212	0.086580	0.082251

Text mining with Description Column - Pipeline

Vectorize Text	Find Correlation between Vector column and target var	Inspect and Find Top predictors
- Vectorize text using any technique - Can consider using dimensionality reduction if use count-based methods such as CountVectorizer or TfidfVectorizer	- For each resulting column, calculate its correlation with target variable - Select top correlated columns or use a cutoff (svd.components_)	- Can use selected columns in model and see if there is improvement - Can also inspect and interpret column output to understand why they correlate with target

Text column interpretation using SVD factor loadings

```
(6, -0.15018636924743195)
```

Column 6 has the highest negative correlation with price (target): -0.15

It makes sense that words like luxury would have a negative loading - the higher the coefficient for luxury, the lower coefficient column 6 will be, and a lower number for column 6 is associated with higher price

Words with lowest loading

```
[('center', -0.07272840672472593),
 ('high', -0.07314626498261155),
 ('brownstone', -0.07491977130806952),
 ('studio', -0.0783623473510409),
 ('beautiful', -0.08270145726799635),
 ('luxury', -0.08282724873363825),
 ('garden', -0.08313558721756222),
 ('new york city', -0.08584393989391555),
 ('york city', -0.08585334043852712),
 ('views', -0.09038845236272569),
 ('building', -0.09170300233121975),
 ('modern', -0.09791023397410266),
 ('city', -0.09839316504879354),
 ('prospect park', -0.10030319935963242),
 ('prospect', -0.12129553554707352),
 ('loft', -0.12883929977176176),
 ('new york', -0.15294873544812715),
 ('york', -0.1537952360737561),
 ('new', -0.20245435856556063),
 ('brooklyn', -0.23794178814603853)]
```

For words with highest loadings it makes some sense as well. For example, the word room has a high loading, and if the listing is offering only a room in Upper East Side then the price would be lower than expected

Top words with highest loading

```
[('room', 0.2607489541339511),
 ('east', 0.24162962148163394),
 ('side', 0.18612705415344288),
 ('east side', 0.17726406582542306),
 ('upper', 0.1371591084619654),
 ('upper east', 0.12211210964326326),
 ('upper east side', 0.11695849014243029),
 ('private room', 0.10687356327390302),
 ('central park', 0.10095671998505162),
 ('train', 0.10035358215588855),
 ('min', 0.09912631392764726),
 ('private', 0.09779675306567276),
 ('east village', 0.09697215074117294),
 ('living room', 0.09145435194038558),
 ('lower east', 0.09064199710251106),
 ('walk', 0.08981976663285329),
 ('lower east side', 0.08925237448207403),
 ('shared', 0.08894019228318803),
 ('lower', 0.07814578113567913),
 ('bathroom', 0.07640692185704813)]
```


NEW YORK
TOKYO
SAO PAULO
KYOTO
RIO DE JANEIRO
SANTA TERESSA
SFpark

Q&A