Spring 5-9-2021

Airbnb Price Prediction with Sentiment Classification

Peilu Liu

Airbnb Price Prediction with Sentiment Classification

A Project

Presented to

The Faculty of Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements of the Degree

Master of Science

by

Peilu Liu

May 2021

The Designated Project Committee Approves the Project Titled

Airbnb Price Prediction with Sentiment Classification

by
Peilu Liu

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

San Jose State University

May 2021

Dr. Robert Chun	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science
Gloria Yan, M.S.	Treasury Analyst, Ross Stores, Inc

Abstract

Airbnb is an online platform that provides arrangements for short-term local home renting services. It is a challenging task for the house owner to price a rental home and attract customers. Customers also need to evaluate the price of the rental property based on the listing details. This paper demonstrates several existing Airbnb price prediction models using machine learning and external data to improve the prediction accuracy. It also discusses machine learning and neural network models that are commonly used for price prediction. The goal of this paper is to build a price prediction model using machine learning and sentiment analysis techniques to help hosts and customers to price the house and evaluate the offered price. Other than using only the numerical data to build the model, customer review is another factor that affects the house pricing. Sentiment analysis techniques, such as Textblob, are also introduced in this project. Compared with using the numerical score of sentiment analysis from customer reviews, we classify sentiments into three types: positive, neutral, and negative. We observe that using classified sentiments with Regression Tree provides a more accurate prediction result compared with using numerical sentiment scores.

TABLE OF CONTENTS

1. Introduction 1

2. Related Works 4

2.1 Price Prediction models 5

2.1.1 Linear Regression 6

2.1.2 Regression Tree and Gradient Boosting 8

2.1.3 Linear Model Ridge 10

2.1.4 Linear Lasso 12

2.1.5 Support Vector Regressor 15

2.1.6 Neural Networks 18

2.2 Data Analysis 20

2.3 Sentiment Analysis 22

3. Dataset 23

4. Implementation and Experiment 24

4.1 Environment Setup 24

4.2 Data Preprocessing 25

4.3 Experiments 29

4.3.1 Linear Regression 29

4.3.2 Regression Tree with Gradient Boosting 31

4.3.3 Linear Model Ridge 32

4.3.4 Linear Model Lasso 34

4.3.5 SVM 36

4.3.6 Neural Network 38

4.4 Results 40

5. Conclusion and Future Works 42

6. References 44

LIST OF TABLES

Table 1. Preprocessing Reviews Using Subjectivity Scores *25*

Table 2. Preprocessing Reviews Using Polarity Scores *26*

Table 3. Testing Results of Prediction without Sentiment *41*

Table 4. Testing Results of Prediction Using Sentiment Polarity *41*

Table 5. Testing Results of Prediction Using Sentiment Classification *41*

LIST OF FIGURES

Figure 1. Linear Regression in 2D Space 7

Figure 2. Gradient Boosting Model with Three Weak Learners 9

Figure 3. Linear Lasso (Left) and Linear Ridge (Right) 13

Figure 4. Support Vector Regressor Model without Errors 16

Figure 5. Support Vector Regressor Model with Errors 17

Figure 6. Simple Neural Network 18

Figure 7. Activation Functions 19

Figure 8. Dataset (reviews.csv) 23

Figure 9. Dataset (listings.csv) 23

Figure 10. Preprocessed Dataset (reviews_cleaned_csv) 26

Figure 11. Sentiment Distribution Pie Chart 27

Figure 12. Sentiment Type VS Normalized Average Price 28

Figure 13. Linear Regression Learning Curve 30

Figure 14. Linear Regression Scalability 30

Figure 15. Regression Tree Learning Curve 32

Figure 16. Regression Tree Scalability 32

Figure 17. Linear Model Ridge Learning Curve 33

Figure 18. Linear Model Ridge Scalability 34

Figure 19. Linear Model Lasso Learning Curve 35

Figure 20. Linear Model Lasso Scalability 35

Figure 21. SVM Ridge Learning Curve 37

Figure 22. SVM Ridge Scalability 37

Figure 23. Neural Network MSE 38

Figure 24. Neural Network Loss Curves 39

1. Introduction

Airbnb is an online platform that provides listing and arrangement for short-term local home renting services. Since its establishment in 2008, it offers 7 million homes and rooms in more than 81,000 cities throughout 191 countries. In 2020, Paris, New York, and London have the greatest number of listings in the world. Nowadays, Airbnb has been the first choice for individuals and groups who are looking for lodging service and tourism experience other than hotels. In the United States, San Francisco, as a global city, is one of the cities that has the greatest number of listings in the state, and is a frequent destination search for Airbnb.

This paper focuses on price prediction using the most recent (Sep 2020) Airbnb data in the San Francisco area. Airbnb price prediction is a useful and important task for guests and hosts. In San Francisco, Airbnb service is provided everywhere, especially in the downtown area. Guests usually evaluate and compare offered prices depending on the home details provided by hosts, such as location, neighborhoods, numbers of bedrooms and bathrooms. The price for each listing is always changing, and the host needs to adjust the price to attract more guests to book the house. The purpose of this project is to develop a suitable model to predict the appropriate listing price using machine learning and neural networks and help customers and hosts to evaluate the price. Models include Linear Regression, Regression Tree, Linear Model Ridge, Linear Model Lasso, SVR, and Neural Network.

Numerical data, such as the number of bedrooms and square footage of the living area, are frequently used for price prediction. However, customer reviews are also one of the key factors that determine the price of each listing. Knowing the feedback from other guests by reading customer reviews provides customers with the first impression of the home. The house that received many positive reviews also deserves higher pricing. Therefore, in this project, each model uses not only the numerical data from the dataset, but also the customer reviews associated with the listing as the feature to build the predictor model. The project uses the open source sentiment analysis technique, Textblob, to score and classify the sentiment of each customer review. The relationship between the sentiment and price would help us to improve the price prediction accuracy and reduce the error. The guidelines to use sentiment analysis to improve the model are described below:

1. Ignore irrelevant comments since they don't give public opinion of the listing to help us predict the price. For example, "Bedrooms are clean. We had a wonderful night.", gives a positive sentiment since "clean" and "wonderful" are positive words. However, "the house is located in the downtown" does not give a public opinion, but just tells a fact. We cannot predict if the customer is satisfied with the house.
2. There are several automatic comments that are generated by the system, such as "The host canceled this reservation 7 days before arrival. This is an automated posting". Some listings contain only automatic

messages, and they are useless in our model. This type of automatic comment is ignored in the data pre-processing step.

3. We notice that negative reviews play a more significant role than positive reviews. Suppose a house has 15 positive reviews talking about the great location and nice neighborhood, and 2 negative reviews talking about dirty bedrooms. Although the amount of positive reviews is greater than negative reviews, negative reviews may prevent customers from making reservations.
4. Other than using the mean of the final sentiment score of each listing directly, we classify it into 3 sentiments: positive, neutral, and negative. The sentiment score tells us how positive or negative overall comments are for a listing. However, we only need to know the overall sentiment of each listing. For example, a sentiment score of 0.5 and 0.7 both represents positive comments. Thus, we classify the sentiment of each listing based on the mean of sentiment scores from comments.

2. Related Works

Based on the large number of public datasets provided by Airbnb, Airbnb price prediction with machine learning technique becomes a popular study. Yuanhang Luo, Xuanyu, Zhou, and Yulian Zhou [1] developed an Airbnb price prediction model using Random Forest, XGBoost, and Neural Network. They eliminated several features, such as host_id and customer_name to reduce noise and keep features like country_code and number of bedrooms to build the model. Textual data, such as house description and neighberhood_review, is also considered as textual features. After performing extensive feature engineering and extraction on the New York and Paris dataset, XGBoost and Neural Network perform better than other models. R-Squared and Median Squared Error (MSE) are used to examine the result. The R-Squared error for XGBoost is 0.722, and 0.769 for Neural Networks.

Another notable implementation of Airbnb price prediction used not only textual data, but also images as features to build the model. Emily Tang and Kunal Sangani[2] used a dataset of listings posted in San Francisco that contains images. They extract multiple sets of features: selected listing information, multinomial bag of words features, text sentiment features, and visual features. One innovative idea is that they extracted visual features from listing images using a bag of words model. Speeded Up Robust Features (SURF) is used to extract descriptors from images. They built a visual word dictionary and created feature vectors for each listing using the descriptor from the images. They only applied Support Vector Machine (SVM) with a linear kernel to analyze the performance and accuracy with each feature used. The accuracy with

sentiment analysis is 0.5499, and listing information is 0.8101. One new idea of this project is that we build the prediction model using the combination of the listing information and sentiment analysis as features to improve the performance.

Other related works considered additional features using external resources. Longitude information and zip code are usually eliminated in most studies since they don't affect the prediction accuracy significantly. However, QuangTrungNguyen [3] computes the number of closest tourist attractions using the location information to improve the performance. All three studies mentioned above used extra features such as textual data, images, and location, other than using only numerical data from the dataset.

We believe that customer reviews are also an important factor that is helpful to build the price predictor. In this project, we analyze and classify customer reviews for each listing as a feature to improve the prediction accuracy. Chapter two and three introduce machine learning models and sentiment analysis techniques that are used in this project and other price prediction studies.

2.1 Price Prediction Models

Machine learning and neural network technologies build models to learn from data automatically without human intervention. They have been used in many different fields such as financial market analysis, natural language processing, image recognition, recommendation systems, DNA sequence, Brain-machine interfaces, etc. In this project, we choose Linear Regression, Gradient Boost, Linear Model Ridge, Linear Model Lasso, SVM, and a simple Neural Network as models to build the price predictor. Each

of the models is designed in different structures and produces different predictions. We introduce machine learning and neural network models in Chapter Two and compare their performance in Chapter Five.

2.1.1 Linear Regression

Linear regression is a machine learning model that models the relationship between a dependent variable and independent (predictor) variable [4]. The model is able to estimate unknown parameters from the data and use linear predictor functions to model the relationship. Linear regression examines if the independent variables are significant to predict the price. For example, the bigger the house, the higher price. However, not all relationships are linear, such as the relationship between longitude and price. Linear relationship exists not only in two-dimensional space, but also multi-dimensional space. If two or more variables are in a linear relationship, a linear regression model can be used to predict the price based on the historical data.

Given two datasets, X: $\{X_1, X_2, X_3, X_4, X_5\}$ and Y: $\{Y_1, Y_2, Y_3, Y_4, Y_5\}$. Values in X and Y represent the input (dependent) variable and the output (independent) variable that the model is trying to predict. In our case, we can assume that X is the area of houses or the number of bedrooms, and Y is the price of the corresponding house. Suppose that each value in X is getting larger for each value in Y, then it implies that X and Y may have a linear relationship, and linear regression is the appropriate model to predict the value of Y. Figure 1 demonstrates the linear regression model in a two-dimensional space graphically. Based on the blue dots, we observe that the larger the

value of X, the larger the value of Y, and it implies that X and Y have a linear relationship. The model goes through the data points in the training set and looks for the red line that is used to predict the value of Y.

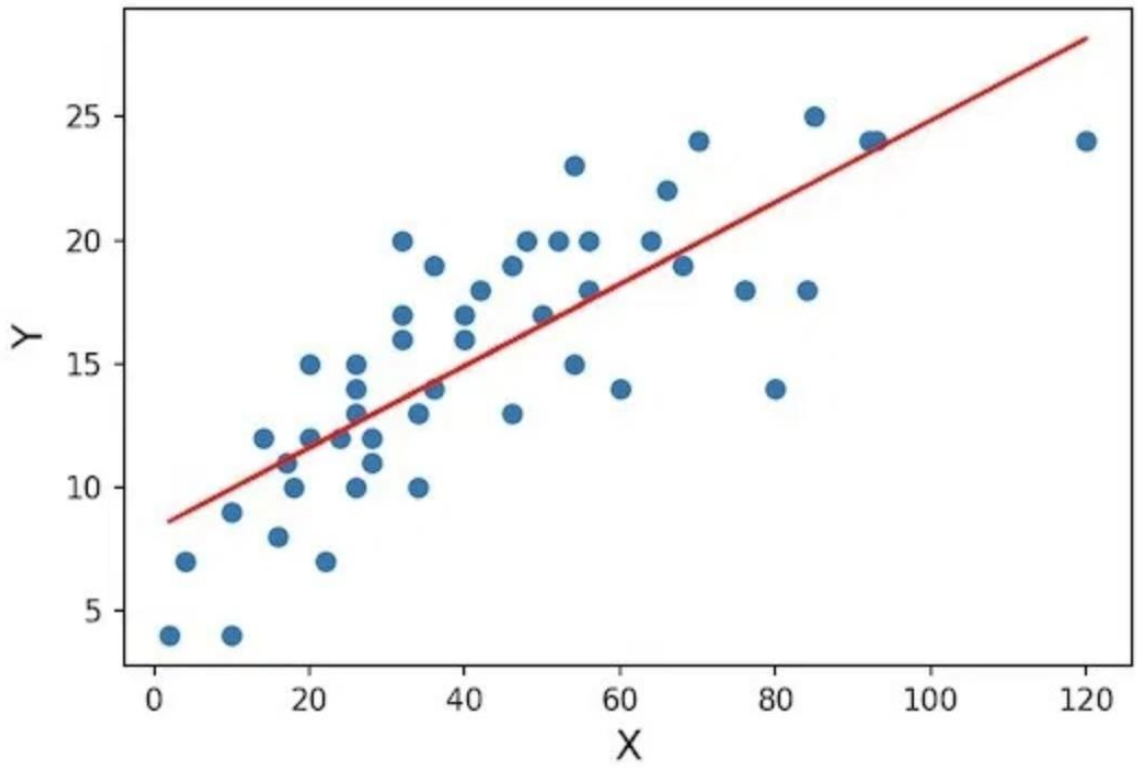

Figure 1. Linear Regression in 2D Space [4]

For simple linear regression, our data is modeled as y = B0 + B1 * X. B0 and B1 represent coefficients that estimate how the line is moved around. B0 is the bias and it determines where the y-axis is intercepted by the line.

The formula of B0 is

$$B0 = mean(Y) - B1 * mean(X)$$

B1 is the slope and it is computed as

$$\Sigma((X_i - mean(X) * (Y_i - mean(Y)) / \Sigma(X_i - mean(X))^2$$

Next, we go through the dataset to estimate the slope based on the formula of B1 and the intercept following the formula of B0. As long as we computed B0 and B1, the model is able to make predictions based on the formula of Y.

Linear regression is used in a wide range of economic applications. It is a supervised machine-learning model that is relatively simple to implement and interpret output coefficients. Compared to other models, if we know there are linear relationships between variables, linear regression is the best choice due to its simplicity. However, outliers in the dataset can affect the regression. The model assumes there is a linear relationship between variables. If there are many outliers, the model may not find the line accurately. In this project, we drop features that do not have a significant effect on price prediction. We assume that there is a linear relationship between the price and customer reviews, house area, number of bathrooms and bedrooms, amenities and score of cleanliness etc.

2.1.2 Regression Tree and Gradient Boosting

Gradient boosting is a technique for classification and regression problems that builds a model by combining a group of weak prediction models, such as regression trees and decision trees. The gradient boosting technique is generalized in a stage-wise fashion by learning and optimizing a differentiable loss function. Leo Breiman [5]

mentioned that the boosting is able to be interpreted as optimization algorithms to appreciate the cost function. Peter Bartlett and Marcus Frean [6] also state that the basic logic of boosting is an iterative functional gradient descent algorithm.

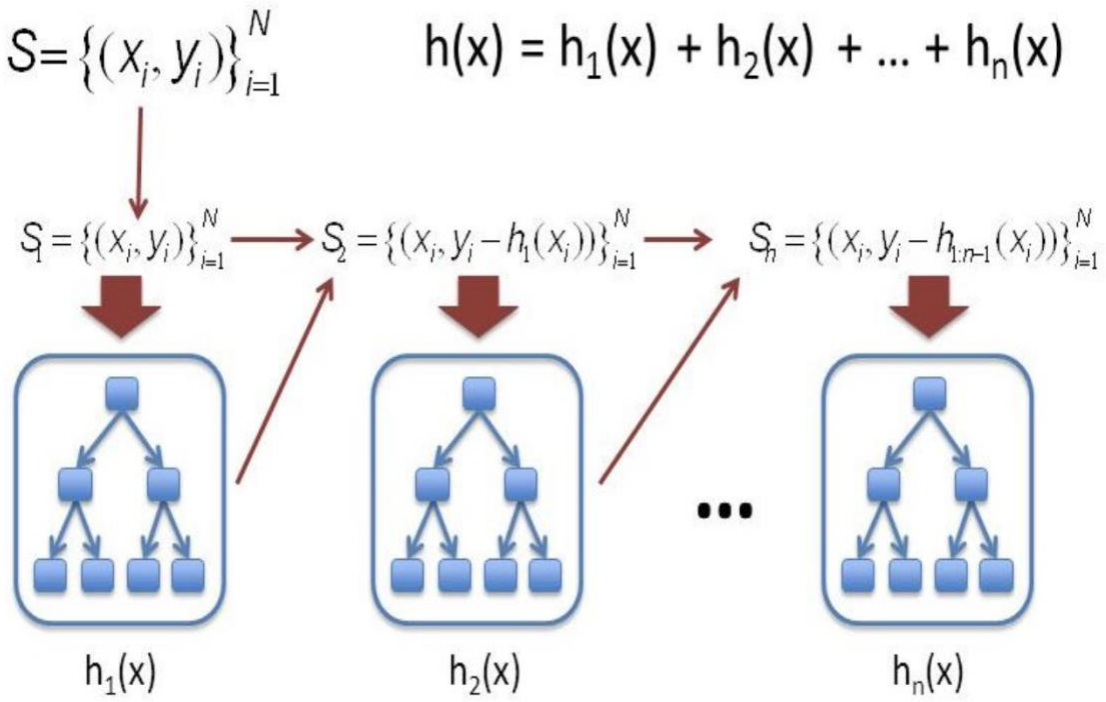

Figure 2. Gradient Boosting Model with Three Weak Learners [7]

For example, in Figure 2, suppose there is a gradient boosting model that is generated by three weak learners (eg. regression tree). If the ground truth value of a sample is 20, and the first weak learner gives 15 as the fitting result, that means the loss is 20 - 15 = 5. In this case, 5 is the fitting goal for the next learner. If the fitting result from the next learner is 12, then the prediction from the boosting model is 12 + 15 = 27. The boosting algorithm optimizes the cost function by choosing a weak hypothesis iteratively that points in the negative gradient direction.

Gradient boosting provides more accurate and fast prediction compared with other models because it allows each stage to make predictions sequentially. In this project, we normalized and pre-processed all features into numerical values. Some values are missing in the dataset, such as when the customer didn't leave a comment. Gradient boosting is able to handle missing data since imputation is not required. However, gradient boosting may be computationally expensive and memory exhaustive if the dataset contains a large number of features and requires many trees. If there are more than 1,000 features in the dataset, sequential processing is mandatory to reduce the computation time. In the data pre-processing step in our project, we drop less significant features such as host_name and keep only 17 columns to increase the computation efficiency. Gradient boosting is also flexible since it allows us to test and optimize on different loss functions. Least absolute deviation, least squares regression and quantile regression are used as loss functions in this project.

2.1.3 Linear Model Ridge

Ridge regression, also named as Tikhonov Regularization, is used to solve the multicollinearity problem in linear regression. The multicollinearity problem occurs when a predictor variable is predicted from other predictors. Although it does not reduce the prediction accuracy of a group of predictors, it may not give a valid result and weight of an individual predictor.

The multicollinearity issue usually happens when the model has a large number of features. When using the least squares method to compute weights of parameters in

the multicollinearity case, values of weights will be very large. The formula for normal linear regression is:

$$y = w^T x$$

Suppose the value of w is large, then the result of y is very sensitive with the value of x. A small change of the value of x may lead to a big change of the result. In general, ridge regression uses the bias-variance tradeoff technique to make the w less sensitive to the change of x value by tolerating an amount of bias and decreasing the condition number.

In our project, suppose we only consider two features: the house area and the number of bedrooms. Let x_1, x_2 and y represent the house area, number of bedrooms, and the price of the house, and we have the formula:

$$\hat{y} = ax_1 + bx_2 + c$$

Since the house area and the number of bedrooms is related with each other, they can cancel each other out. The solution is to set *a* to a large positive number and *b* to a negative number that has a large absolute value. However, this may cause values of *a* and *b* to become non-sensible and reduce the interpretability of the model. Linear ridge regression gives a constraint to the model, such as $a^2 + b^2 \leq t$. The formula of least squares method is:

$$f(w) = \sum_{i=1}^{m} (y_i - x_i^T w)^2$$

When multicollinearity occurs, the results of the least squares method become unstable. The solution to prevent the multicollinearity issue is to add subject to the constraint and limit the value of w. The final formula is demonstrated as a Lagrangian:

$$f(w) = \sum_{i=1}^{m} (y_i - x_i^T w)^2 + \lambda \sum_{i=1}^{n} w_i^2$$

2.1.4 Linear Lasso

Lasso (Least Absolute Shrinkage and Selection Operator) is a machine learning technique that enhances the interpretability and prediction accuracy by performing feature selection and regularization. Instead of using all covariates in the final model, Lasso changes the process of model fitting and selects only a subset of covariates. Same with linear ridge, Lasso regression modified the cost function to reduce the sensitivity of certain features. The difference between linear ridge and lasso is that linear ridge uses the L2 penalty to prevent overfitting while Lasso uses the L1 penalty to select features that have non-zero coefficients. After adding the L1 regularization, the cost function for Linear Lasso is:

$$J = \frac{1}{n}\sum_{i=1}^{n}\left(f\left(\boldsymbol{x}i\right) - yi\right)^2 + \lambda\left\|w\right\|_1$$

Compared with linear ridge, linear lasso can be used for feature selection. The basic idea of lasso is to set a constraint and force the absolute value of coefficients to be less than a fixed value. This is similar to the linear ridge that adds a constraint to increase the distance between coefficients. However, linear lasso forces certain coefficients to be zero and eliminate the corresponding features in the model.

Figure 3 shows the estimation graph for the linear lasso (left) and linear ridge (right). The red ellipses represent the least square error function, centered at the same location. The blue areas indicate the constraint area, $|\beta_1| + |\beta_2| \leq t$and ${\beta_1}^2 + {\beta_2}^2 \leq t^2$, respectively. In the linear lasso graph, we can only choose points in the blue rectangle. The final prediction of linear lasso is the intersection of the red ellipse and blue rectangle. In the graph, the intersection is located at the vertex of the blue rectangle. At this point, the coefficient of the corresponding feature becomes zero and the feature is eliminated from the model. In the linear ridge graph, the

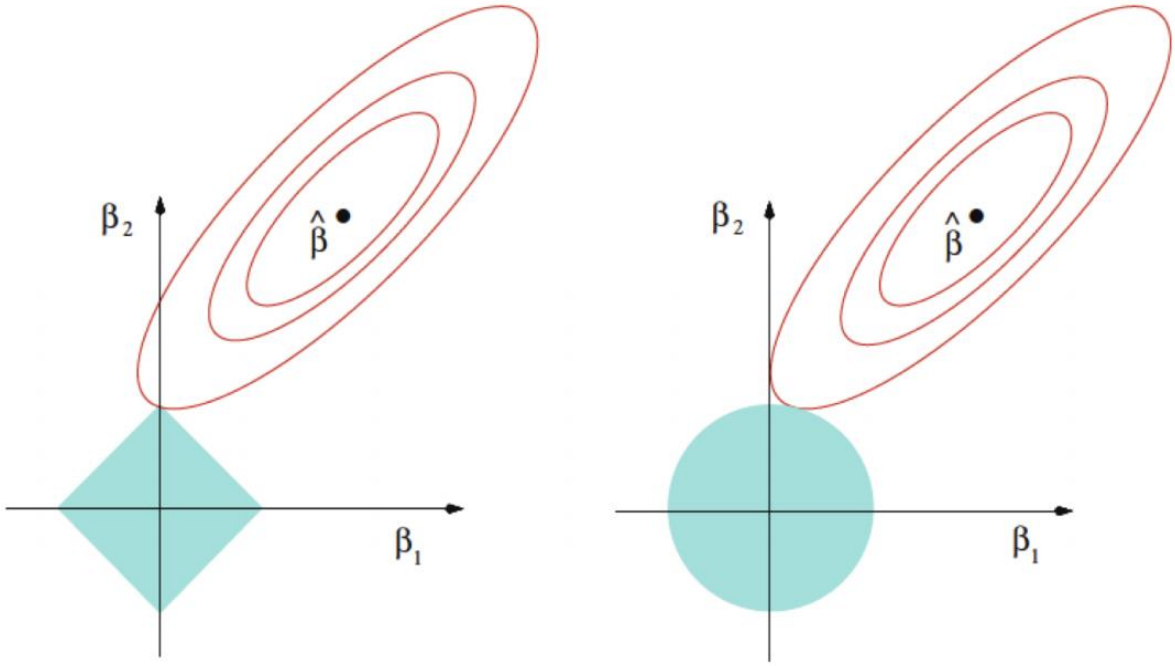

Figure 3. Linear Lasso (Left) and Linear Ridge (Right) [8]

intersection is located at the tangent point between the eclipse and the blue circle. They are not able to intersect at the y-axis and this makes ridge regression not able to update the coefficient to zero.

In this project, we first manually select 17 features, including the customer review, that we think are important for the price prediction model. The linear lasso model automatically eliminates the features that are less important by updating the coefficient to zero. It shows one advantage of linear lasso is that when we don't have sufficient samples for certain features, linear lasso helps us to eliminate the feature and reduce the mean square error. For example, if the dataset does not have enough information in the "bathroom_text" feature, this feature will be removed from our

model. In other words, only the features that actually affect the price prediction are kept for building the model.

2.1.5 Support Vector Regressor

Support Vector Regression (SVR) is a regressor that predicts continuous variables. It uses similar algorithms with Support Vector Machine (SVM), but SVM performs classification and predicts discrete labels. The subject in most regression models is to minimize the squared errors, such as ordinary least squares. The OLS is shown as below, where y_1, w_i, and x_i represent the prediction result, coefficient, and feature, respectively:

$$MIN \sum_{i=1}^{n} (y_i - w_i x_i)^2$$

In section 2.1.3 and 2.1.4, linear ridge and linear lasso are extended from this form of linear regression with constraints and penalty to improve the accuracy and reduce the complexity and number of less important features.

SVR finds a line in two dimensions or hyperplane in high dimensions and to fit the data. We define a variable to represent the number of errors that the model can tolerate. The objective and constraint functions for SVR are shown below:

Objective Function:

$$MIN \ \frac{1}{2} \left|\left|\boldsymbol{w}\right|\right|^2$$

Constraint Function:

$$|y_i - w_i x_i| \leq \varepsilon$$

The objective function is to minimize the L2 normalization of the coefficient vector. Compared with OLS in normal linear regression, the constraint function handles the error (ε in the constraint function). The error is the absolute error located in a specified margin which is defined by the constraint function.

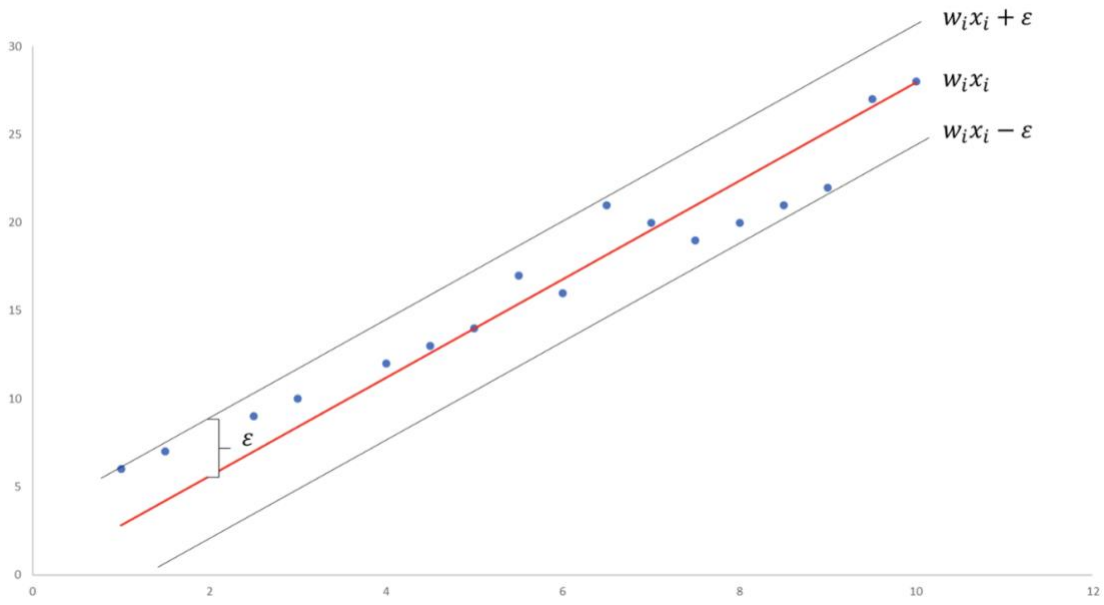

Figure 4 Support Vector Regressor Model without Errors [9]

Figure 4 shows a simple illustrative example of SVR. The red line demonstrates the line that best fits the dataset and the two grey lines show the specified margin of ε. The error, ε, is defined by the developer. In this example, all data points are perfectly located in the margins. In our project, not all data points are located in the specified margin.

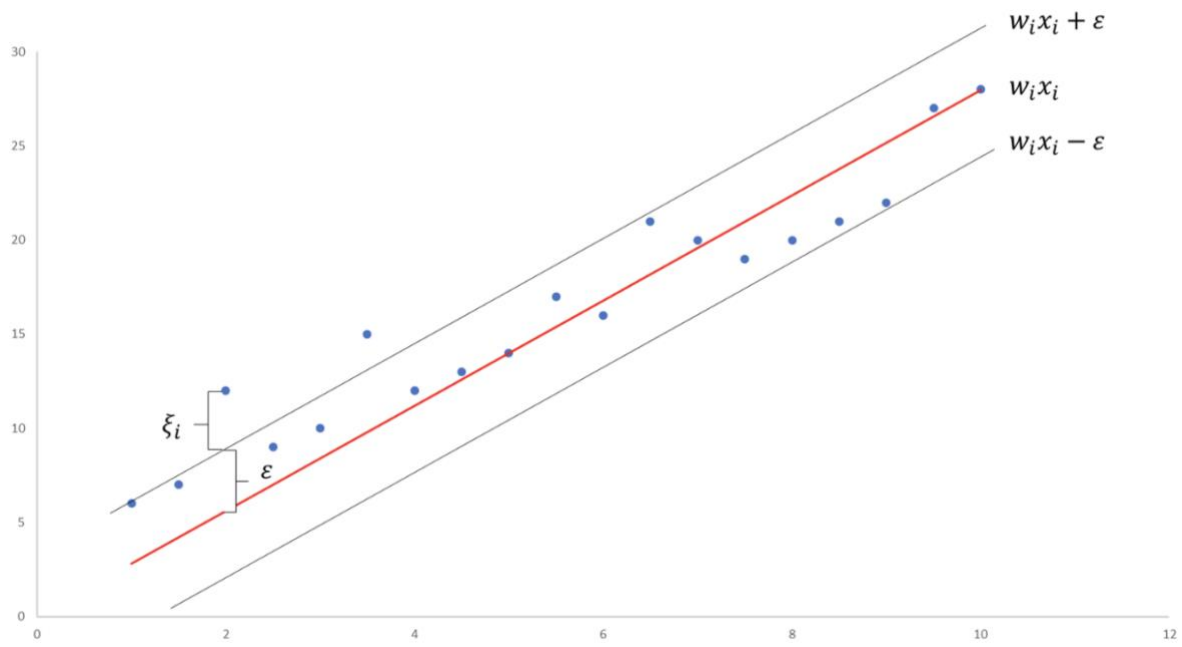

Figure 5 Support Vector Regressor Model with Errors [9]

As shown in Figure 5, we need to allow a certain number of errors that are located outside of the margin. In this case, slack variables are needed to denote the deviation from the specified margin as ξ. The updated objective and constraint functions are shown below:

Objective Function:

$$MIN\ \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}|\xi_i|$$

Constrain Function:

$$|y_i - w_i x_i| \leq \varepsilon + |\xi_i|$$

Overall, SVR is a powerful and flexible regression model since it allows the scientist to choose the way to tolerate errors. The scientist is able to build and test the SVR model with different values of threshold (ε) and margin of tolerance (ξ). The margin of tolerance allows us only to take points that are within the boundary. It helps us to build a better fitting model with the least error rate.

2.1.6 Neural Networks

Neural networks are algorithms or a circuit of neurons for solving artificial intelligence problems. They have been used in many applications such as price prediction, weather forecasting, financial services, marketing research, fraud detection, face detection and recognition, and natural language processing, etc. The structure of a simple neural network is shown in Figure 6:

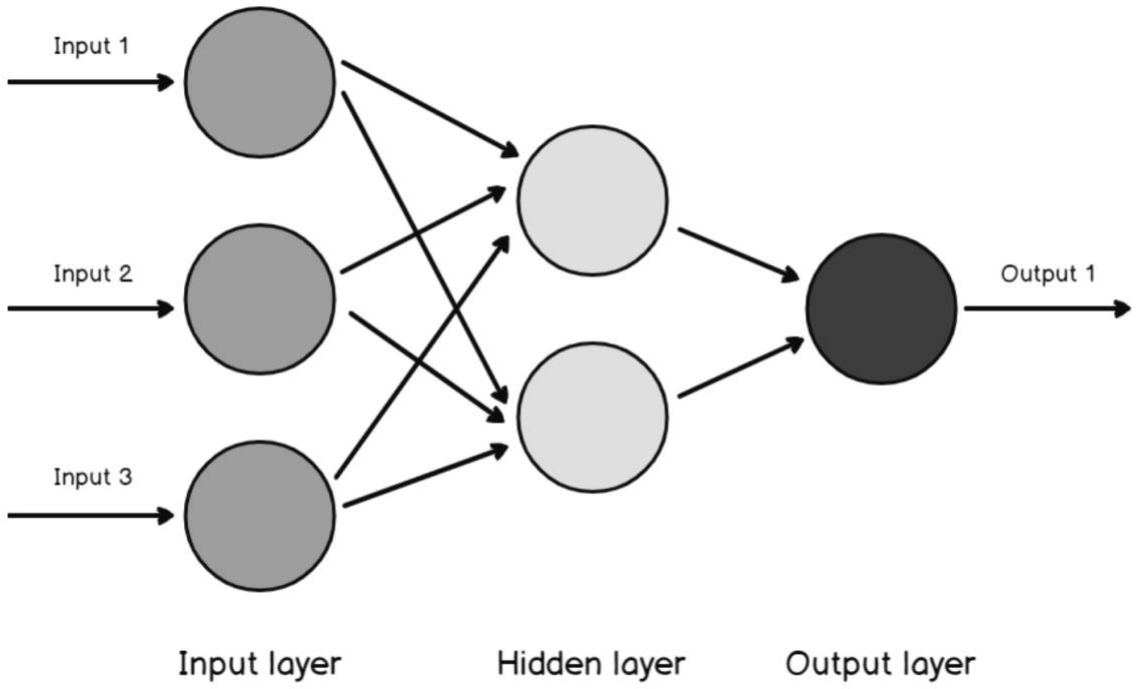

Figure 6 Simple Neural Network [10]

A neural network includes an input layer, several hidden layers, an output layer and nodes. Each node is interconnected with each other to feed the data generated by linear regressors to the activation function. The hidden layer fine-tunes weights from the input until the error margin of the neural network is minimized. This process is also called feature extraction, which is similar with principal component analysis. The output of each node is defined by the given input and the activation function.

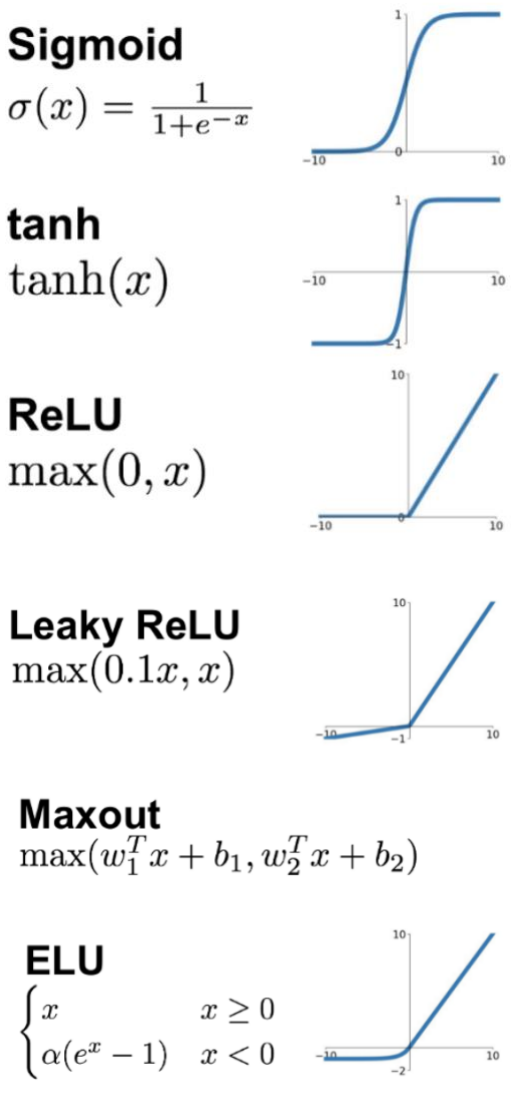

Figure 7. Activation Functions [11]

Figure 7 demonstrates existing activation functions that are commonly used in neural networks. In this project, we implement and test our neural network with Sigmoid and ReLU. Sigmoid function is ranged from zero to one, and not zero centered. However, it contains an exponential operation which might be computationally expensive. ReLU is the activation function that is commonly used in neural networks. Compared with other activation functions, ReLU is not time-consuming during backpropagation and not computationally expensive. Since the objective of this project is to compare the performance of different machine learning models and neural networks, we only test the performance of the neural network with Sigmoid and ReLU and leave other activation functions for future implementations.

2.2 Data Analysis

Large datasets usually contain a great number of features, but not all of them are needed for price prediction. For example, host_identity_verified has less affection than the number of bedrooms. Using unneeded features may require heavy computation and reduce the prediction accuracy by producing noise during the training process [12]. There are several existing solutions to reduce dimensionality and keep the most meaningful features. Principle Component Analysis is a commonly used linear technique to reduce dimensionality by mapping data to a lower dimensional space linearly [13]. Generalized Discriminant Analysis (GDA) and Uniform Manifold Approximation and Projection (UMAP) are two nonlinear techniques for dimensionality reduction.

In our project, we are interested in knowing which features are more significant for price prediction by calculating the p-value. The p-value represents the significance of a certain property under the assumption of a null hypothesis [14]. A large p-value indicates that the given property is closely related to the assumption, while a small p-value shows the outcome is unlikely affecting the observation. We setup a threshold to eliminate unneeded features and keep a certain number of significant features. Sklearn provides us a scoring function for feature selection by computing the p-value for all features in the dataset. First, we compute the covariance of the input (x) and the expected result (y) following the formula:

$$Cov\left(x, y\right) = \frac{\Sigma(X-\bar{X})(Y-\bar{Y})}{n-1}$$

Next, we compute the correlation coefficient to get the significance of the given property:

$$r = \frac{Cov(x,y)}{\sqrt{s_x^2 s_y^2}}$$

Once all p-values are computed, we compare each p-value with the threshold: features with p-values greater than the threshold are kept for training, while the rest are eliminated. This way, the training time for each model is decreased and the prediction accuracy is improved by reducing the noise.

2.3 Sentiment Analysis

The sentiment analysis technique has been used in natural language processing to extract and study emotions from human languages for years. One example of sentiment analysis is to effectively analyze and extract the opinion from customer reviews of a product [15]. For instance, “this cell phone has a great camera, but the battery is bad”, demonstrates a positive opinion of the camera and a negative opinion of the battery. Business owners analyze opinions of different aspects to gather feedback from customer reviews efficiently.

Airbnb allows customers to leave reviews of their visits, and we believe that the review is a factor that affects the price of each listing. The house owners adjust the price based on the feedback from customers, while the customer evaluates the price based on previous customer reviews. In our project, we use TextBlob [16] to analyze and classify reviews into three categories: positive, neutral, and negative based on the sentence polarity. Additionally, some reviews, such as “The house is located in a forest”, are just stating a fact, other than giving an opinion. We get rid of these types of sentences to reduce the noise based on their subjectivity since they are not helpful to improve the prediction accuracy.

3. Dataset

We choose the dataset provided by Inside Airbnb [17] that consists of general information of listings and reviews for different cities. In this project, we use two split datasets that contain the booking information up to September 2020 for San Francisco. The first dataset, “listings.csv”, describes the basic information of each listing, such as house id, host location, accommodations, number of bedrooms, etc. Some of the information doesn't affect our prediction accuracy such as listing URL, license, or host name. They are eliminated in the training phase to reduce the noise. The second dataset, “reviews.csv”, contains historical customer reviews for each listing id. Customer reviews are evaluated and used as a feature for each listing id. Figure 8 demonstrates the customer reviews for the listing with the listing_id, 958. Each listing has around 5 to 150 customer reviews. Figure 9 indicates partial listing details and price are provided in the “listings.csv”. The “listings.csv” contains 7054 distinct listings and 75 features for each listing id.

	listing_id	id	date	reviewer_id	reviewer_na	comments
2	958	5977	7/23/09	15695	Edmund C	Our
3	958	6660	8/3/09	26145	Simon	Returning to San Francisco is a rejuvenating thrill but this time it was enhanced by our stay at Holly and David's beautifully renovated and perfectly located apart
4	958	11519	9/27/09	25839	Denis	We were very pleased with the accommodations and the friendly neighborhood. Being able to make a second bed out of the futon couch was particularly helpfu
5	958	16282	11/5/09	33750	Anna	We highly recommend this accomodation and agree with the previous postings: Holly and David were extremely helpful and friendly (but not at all instrusive), th
6	958	26008	2/13/10	15416	Venetia	Holly's place was great. It was exactly what I needed. Perfect location, super clean, and even a little patio out back. Since I go back to San Francisco frequently, t
7	958	29240	3/13/10	78623	Mathieu	On top of all
8	958	32988	4/3/10	96027	Lauren	This apartment was everything I could have hoped for and more. The place is extremely well appointed, comfortable and situated in one of the best places to be
9	958	220443	4/8/11	426888	Michelle	Great place
10	958	567690	9/26/11	539189	Sarah	Great location in San Francisco, really felt like a "local" rather than a "tourist" with super restaurants and transport links within a short walking distance. Apartm
11	958	1786860	7/24/12	2059845	JÃ¶rg	We had a
12	958	2214082	9/6/12	2869693	Carrie	Holly's unit is comfortable and convenient if you wish to stay in the Haight neighborhood. We had privacy, yet Holly was right there if we needed anything. My
13	958	2263537	9/11/12	2953444	Adrienne	Holly's place
14	958	3553733	2/16/13	134792	Robert	This is a
15	958	4392049	5/2/13	1635153	Lisa	Our stay at Holly's apartment was absolutely wonderful. The neighborhood is beautiful and the apartment is very close to transportation. The apartment was th
16	958	5040421	6/9/13	6413536	Andrew	Great location near Haight Street and close to Muni to get into downtown. Friendly hosts who helped with sunscreen, lost luggage and parcel delivery. Very rel
17	958	5171535	6/16/13	5975179	Boaz	Truly amazing place - great location near Haight Street, close to Muni station, 100 feet from a lovely park. The house itself is very clean, lovely decorated, with fu
18	958	5267286	6/21/13	4549217	Cathy	We really enjoyed staying in this cute, and airy garden apartment. It was a little cramped for me and my 2 teen boys, but we had everything we needed. The kitc
19	958	5905160	7/22/13	5685916	Isabelle	This was the perfect little rental suite. It was equipped with everything you need including cookware. The place was clean and very cute. A nice addition was the

Figure 8. Dataset (reviews.csv)

	id	listing_url	scrape_id	last_scraped	name	description	neighborhoo	picture_url	host_id	host_url	host_name	host_since	host_locatio	host_about	host_respon	host_respon	host_accept	host_is_supe	host_thumb	host_picture	host_neighb	host_li
2	958	https://www	2.0201E+13	9/7/20	Bright, Mode	Cleaning Pro	Quiet cul de	https://a0.m	1169	https://www	Holly	7/31/08	San Francisc	We are a fan	within an ho	100%	99%	t	https://a0.m	https://a0.m	Duboce Triar	
3	5858	https://www	2.0201E+13	9/7/20	Creative San	<b>The spac	I love how o	https://a0.m	8904	https://www	Philip And Ta	3/2/09	San Francisc	Philip:	within a day	70%	84%	f	https://a0.m	https://a0.m	Bernal Heigh	
4	7918	https://www	2.0201E+13	9/8/20	A Friendly Rc	Nice and goc	Shopping olc	https://a0.m	21994	https://www	Aaron	6/17/09	San Francisc	7 minutes	within a few	100%	100%	f	https://a0.m	https://a0.m	Cole Valley	
5	8142	https://www	2.0201E+13	9/8/20	Friendly Roo	Nice and good	public trans	https://a0.m	21994	https://www	Aaron	6/17/09	San Francisc	7 minutes	within a few	100%	100%	f	https://a0.m	https://a0.m	Cole Valley	
6	8339	https://www	2.0201E+13	9/8/20	Historic Alam	Pls email before booking.		https://a0.m	24215	https://www	Rosy	7/2/09	San Francisc	I'm an	within a few	100%	0%	f	https://a0.m	https://a0.m	Alamo Squar	
7	8739	https://www	2.0201E+13	9/8/20	Mission Suns	Welcome to	Located betv	https://a0.m	7149	https://www	Ivan & Wend	1/27/09	San Francisc	Ivan is a	within an ho	100%	95%	t	https://a0.m	https://a0.m	Mission Distr	

Figure 9. Dataset (listings.csv)

4. Implementation and Experiment

4.1 Environment Setup

This project is executed and analyzed on the macOS Catalina Version 10.15 operating system with a 2.6 GHz Quad-Core Intel Core i7 processor and 16GB 2133 MHz LPDDR3 memory. We implement the project using Python which is a suitable programming language to build machine learning models. Its simplicity allows developers to focus on the model implementation instead of the language. Additionally, Python provides many Deep Learning and Machine Learning frameworks to reduce the implementation time [18]. In this project, we use NumPy, Pandas, Scikit-learn, and Keras for data processing and model implementation. NumPy supports sufficient computation in multi-dimensional matrices and we use NumPy to compute the mean, sum, and standard deviation etc. of each property. Locating and modifying multiple inputs in a large dataset is time consuming. Pandas is the solution that helps us to read and manipulate the input sufficiently. It allows the developer to apply customized lambda functions to specific columns such as replace empty input with the mean. Scikit-learn contains various machine learning models such as linear regression, support vector machines, and gradient boosting. All models in this project, except the neural network model, are implemented using the Scikit-learn framework. The neural network model is built with Keras, which offers the developer the flexibility to setup layers and activation functions with customized learning rates, decay rates, and epochs, etc.

4.2 Data Preprocessing

Our two datasets contain numerical, textual, and empty data, and we need to clean and normalize the data. The data preprocessing task includes five main steps:

(1) Score and classify the sentiment of reviews for each listing.

(2) Remove columns that are not related to our task, such as host email and house URL.

(3) Convert all data types into numbers and fill empty entries with specified values.

(4) Normalize each column and split the data into training set (90%) and testing set (10%).

(5) For comparison purposes, compute p-values to select features that affect the price significantly.

In the first step, we use TextBlob to compute the subjectivity score to take special care of reviews that are just telling a fact. As shown in Table 1, if the subjectivity score is equal and less than 0.75, the review is categorized to a neutral review. Otherwise, we use the polarity score to classify the sentiment of each review. In Table 2, reviews that have a polarity score lower than zero, are classified as negative reviews. Reviews that have polarity scores greater than 0.2 are classified as positive reviews. Other reviews are classified as neutral reviews.

Subjectivity <= 0.75	Subjectivity > 0.75
Stating a Fact	Contains Sentimental Information

Table 1: Preprocessing Reviews Using Subjectivity Scores

Polarity < 0	0 <= Polarity < 0.2	Polarity >= 0.2
Negative	Neutral	Positive

Table 2: Preprocessing Reviews Using Polarity Scores

After classifying the sentiment, we read the data of each listing from listing.csv. Each row indicates a listing by its unique ID and each column represents a feature. To reduce noise and improve prediction performance, features that are not related to price prediction are eliminated. Additionally, some features are in text form which are not readable for the chosen python package, pandas. Therefore, in this step, we convert all features into numerical data. For example, the feature, instant_bookable, has only two options: “yes” or “no”. This type of feature is converted to 1 or 0, respectively. Since not all inputs are in the same scale, we normalize each input’s range from 0 to 1. Some features are empty in the dataset and cause exceptions in run-time. First, we assume the missing input represents a “no” response and fill all missing inputs with 0. However, missing inputs in numerical features, such as “minimum_nights”, are just not provided by hosts. In this case, we are not able to assume the “minimum_night” for the missing input is “yes” or “no”. Finally, we decide to replace all missing inputs with the average of the whole column. The preprocessed and normalized dataset is demonstrated in Figure 10.

1	id	host_id	host_since	host_respon	host_is_supe	host_total_li	host_has_pr	host_identity	latitude	longitude	accommodat	bedrooms	beds	price	minimum_ni	maximum_n	minimum_m	maximum_m	minimum_m	maximum_m	minimum_ni
2	958	1169	3754	100	1	1	1	1	37.76931	-122.43386	3	1	2	4.91265489	2	30	2	2	1125	1125	2
3	5858	8904	3540	70	0	2	1	1	37.74511	-122.42102	5	2	3	5.45958551	30	60	30	30	60	60	30
4	7918	21994	3433	100	0	10	1	0	37.76555	-122.45213	2	1	1	4.02535169	32	60	32	32	60	60	32
5	8142	21994	3433	100	0	10	1	0	37.76555	-122.45213	2	1	1	4.02535169	32	90	32	32	90	90	32
6	8339	24215	3418	100	0	2	1	1	37.77525	-122.43637	4	2	2	6.62804138	5	111	5	5	111	111	5
7	8739	7149	3574	100	1	2	1	1	37.7603	-122.42197	3	1	1	5.12989871	1	14	1	1	14	14	1

Figure 10. Preprocessed Dataset (reviews_cleaned.csv)

In the last step, we compute the p-value [19] for each feature to select the most significant features. After comparing the p-value with the threshold, 1e-20, 13 features are selected, which are accommodates, number of bedrooms, number of beds, calculated_host_listing_count_shared_rooms, entire condominim, entire house, entire loft, private room in house, entire home/apt, private room, shared room. We test the feature selection process with different thresholds and observe that the current threshold gives us a reasonable number of features to use for training. Finally, 90% of the preprocessed dataset is split for training and the remaining 10% is for testing.

After data preprocessing, we notice that the dataset contains more positive reviews than neutral and negative reviews. As shown in Figure 11, more than half of the reviews are positive reviews. Although comments are not equally distributed, we expect that listings with positive reviews have a higher price than listings with neutral and negative reviews.

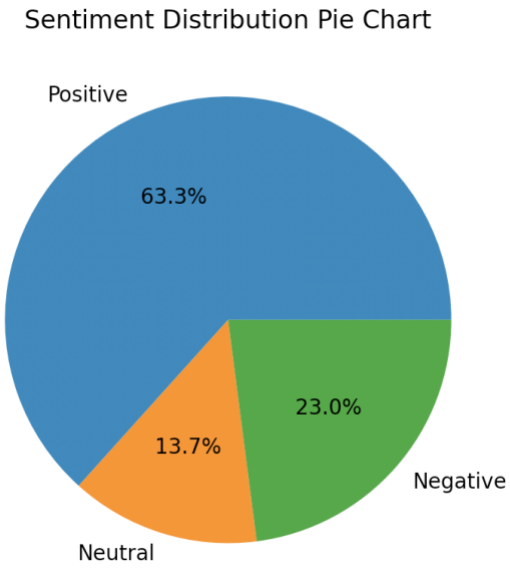

Figure 11. Sentiment Distribution Pie Chart

We compute the average price (normalized) of each sentiment type to understand the significance of sentiment to price prediction. As shown in Figure 12, listings with positive comments have a higher average price than listings with neutral and negative comments. Neutral reviews don't affect the price significantly since they may contain automatic comment or comments that do not tell an opinion. Additionally, we notice that negative reviews significantly decrease the house price. Overall, based on Figure 11 and Figure 12, we expect that using customer reviews with sentiment classification as a feature is indeed able to help us in building a more accurate price prediction model.

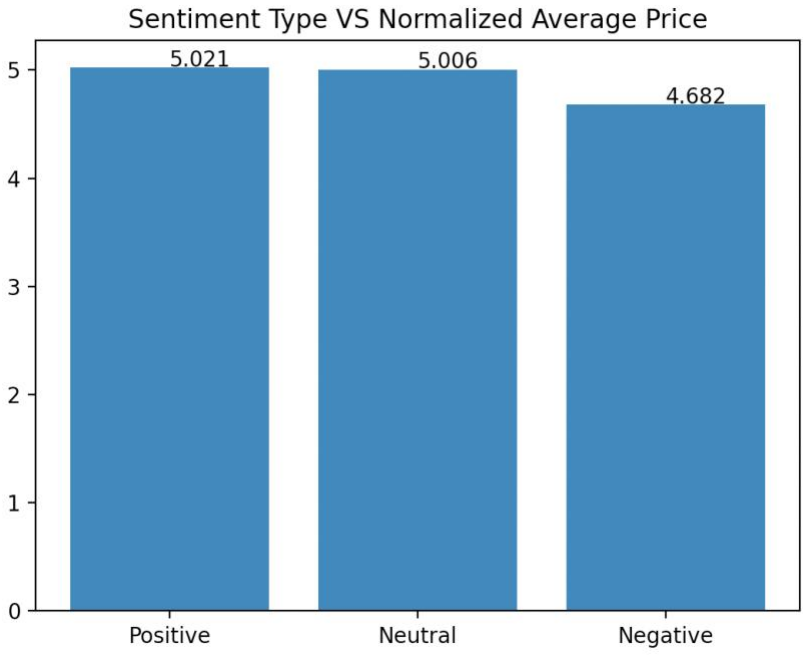

Figure 12. Sentiment Type VS Normalized Average Price

4.3 Experiments

The goal of this project is to select the machine learning model that makes the most accurate prediction on Airbnb house prices. Our proposed prediction method is making price prediction using sentiment classification and subjectivity score. We use the same dataset to compare its performance with two prediction methods from previous researches, which are making prediction without sentiment analysis, and making prediction using sentiment polarity score. Our implementation is an extension of existing project that uses sentiment scores to make price predictions [20].

Six selected machine learning and deep learning models from Chapter 2 are trained and analyzed for each prediction method. We analyze the performance of models using a learning curve, scalability curve, mean squared error curve, and model loss curve from the training history. Finally, we compare its mean absolute error, median absolute error, mean squared error, and R-squared with the two price prediction methods from previous researches.

4.3.1 Linear Regression

We choose the linear regression model as our baseline model. The LinearRegression module from scikit-learn allows us to define several parameters such as fit intercept and normalize; however, only the n_jobs parameter is specified as 0.8 to multiply the number of cores for speedup. Overall, the implementation of Linear regression is the simplest compared with other selected models.

Figure 13 demonstrates the learning curve from the training history. The red line represents the training score while the green line indicates the cross-validation score. We choose R2 as the scoring function which is also the default scoring function for regression models. The training score starts from around 0.6 and decreases to below 0.4. The cross-validation score starts from 0.15 then stabilized around 0.35. Figure 14 indicates the scalability curve of the regression model. The fit-times represent the time spent for fitting in seconds [21]. The fit time grows from 0.0046 second to 0.0064 second as the amount of training examples increases.

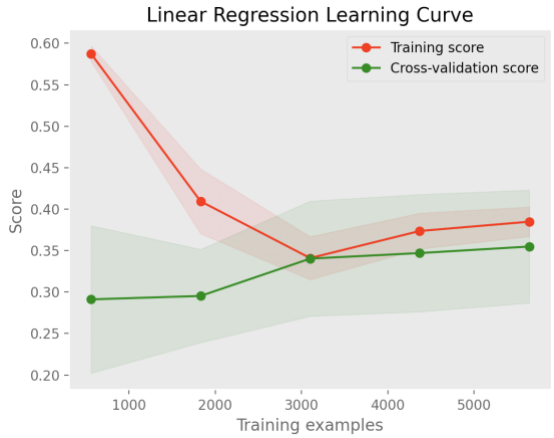

Figure 13. Linear Regression Learning Curve

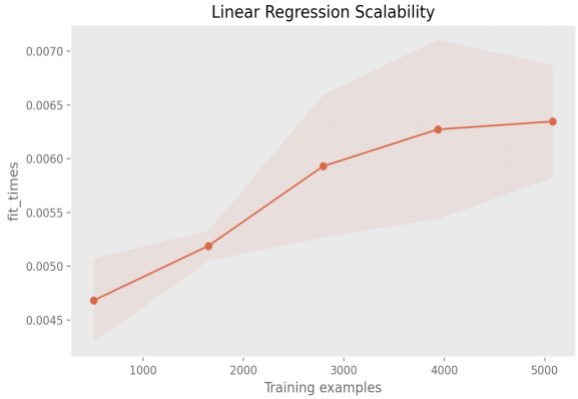

Figure 14. Linear Regression Scalability

4.3.2 Regression Tree with Gradient Boosting

We use the GradientBoostingRegressor from the sklearn package to build the regression tree model. There are 13 boosting stages (n_estimators) to perform because 13 features are selected in the feature selection step. The learning rate and maximum depth are set to 0.12 and 10 respectively. The GradientBoostingRegressor model supports two loss functions: least absolute deviation and least squares regression. We test the performance of the two loss functions and observe that the least squares regression achieves 0.433 as the mean absolute error while the least absolute deviation's mean absolute error is 0.312.

Regression tree with gradient boosting provides the highest training score, 0.77, among all selected models. As shown in Figure 15, the training score is settled to around 0.77 and the cross-validation score is 0.61. Overfitting is observed since the cross-validation score is lower than the training score by 19%. The fit time at 3,000th samples is 0.94 second longer than the linear regression model's fit time. In Figure 16, the fit_times reaches 0.9 second as the training examples reach 5,000. Since the fit time increases linearly, we expect it will continue to increase if there are more training samples.

Figure 15. Regression Tree Learning Curve

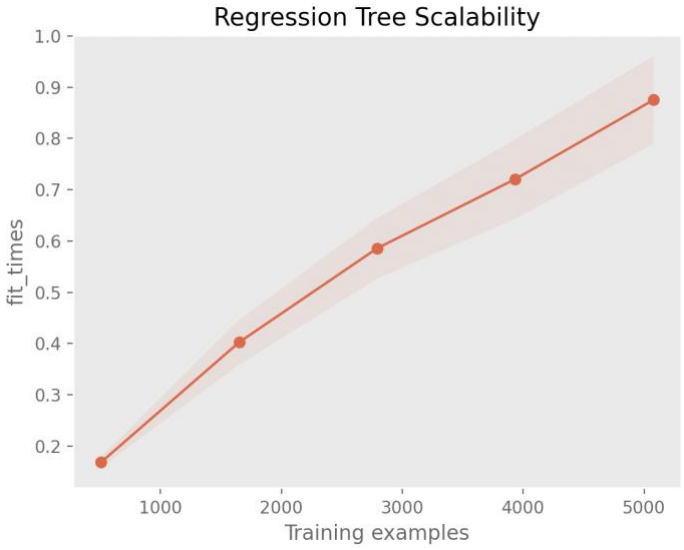

Figure 16. Regression Tree Scalability

4.3.3 Linear Model Ridge

The Ridge classifier is implemented using the linear_model module from scikit-learn, which builds a regression model with linear least squares as the loss function and

L2 regularization. Although the classifier supports several parameters, we only define the alpha parameter to 0.7, since defining other parameters with different values produce the same mean absolute error, 0.454.

Figure 17 indicates that the score of both the training curve and cross-validation curve are settled at around 0.33, which is 63% lower than the training score of regression tree with gradient boosting. However, the score continually increases as the number of samples increases, so we expect that more samples would bring a better score. As shown in Figure 18, the fit time is between 0.0028 second to 0.0031 second, which is around 50% faster than regular linear regression mode.

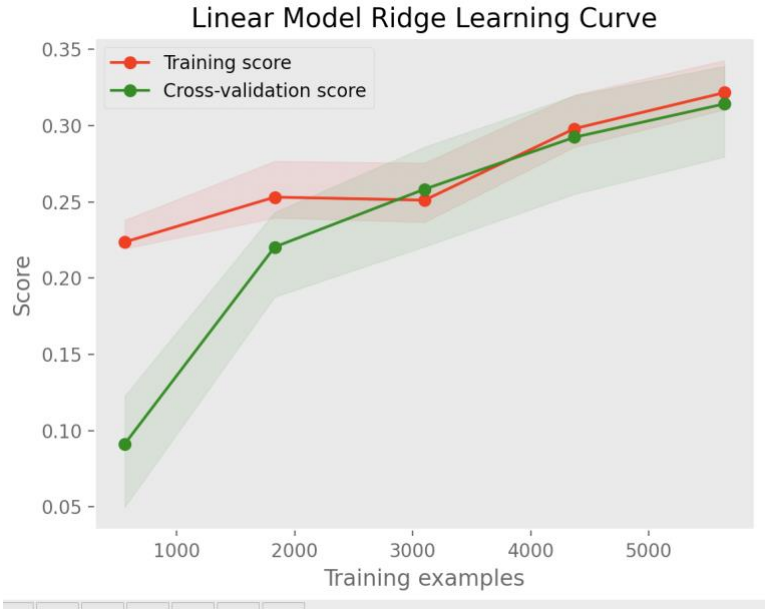

Figure 17. Linear Model Ridge Learning Curve

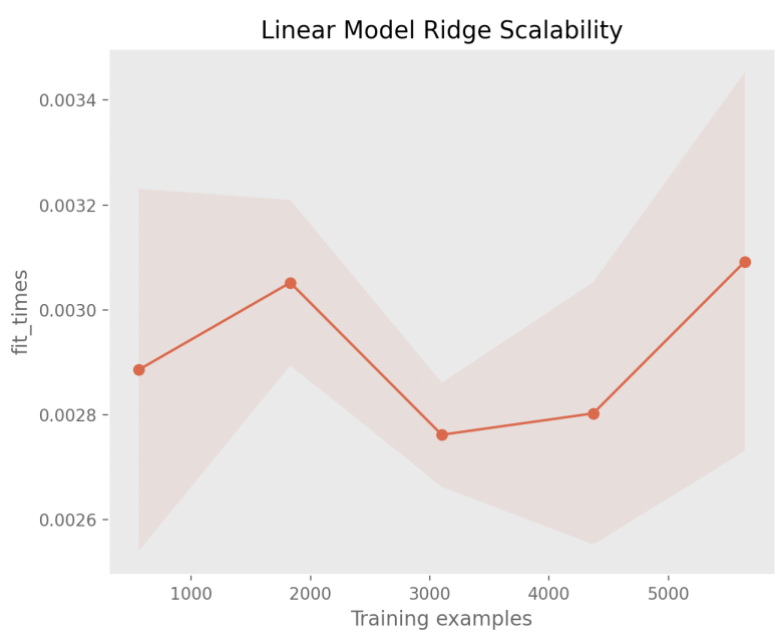

Figure 18. Linear Model Ridge Learning Curve

4.3.4 Linear Model Lasso

We use the Lasso module from the scikit-learn package to build the model. The setup of Linear Lasso is similar with the setup of Linear Ridge in which we only define the "alpha" parameter to 0.5. The "alpha" represents a constant that multiplies the L1 regularization [22]. Other parameters such as fit_intercept and precompute are tested with different values, and we observe that those parameters do not impact the prediction accuracy since they produce the same mean absolute error, 0.609.

In Figure 19, the learning curve is in a similar shape with the linear model learning curve, however, they start and end in different values. The training score starts from around 0.62 and ends at 0.4, while the cross-validation score starts from 0.30 and ends at 0.35. Compared with the Linear Regression model, Linear Lasso's score is 11%

lower in training and cross-validation phases. However, as shown in Figure 20, the fit time of linear lasso, 0.0029 second, is around 200% lower than the Linear Regression model, 0.0065 second, and the same with the linear ridge model, 0.0023 second. Overall, the average training score of Linear Lasso, 0.42, and its fit time, 0.0029 second, is at the average and the shortest among other models, respectively.

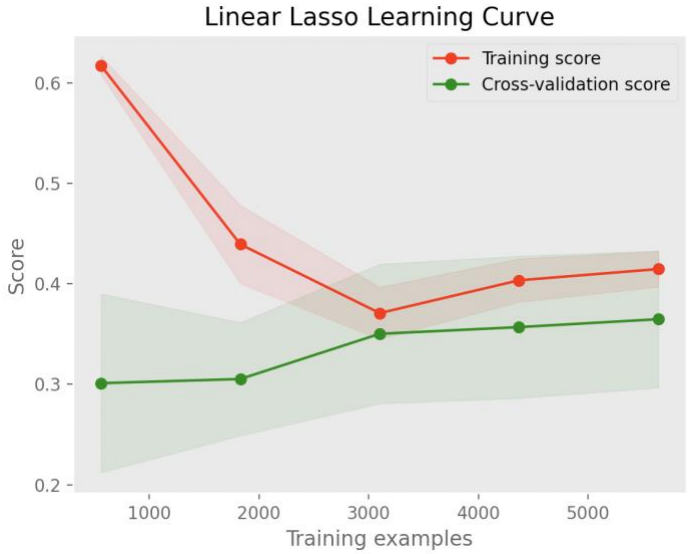

Figure 19. Linear Lasso Learning Curve

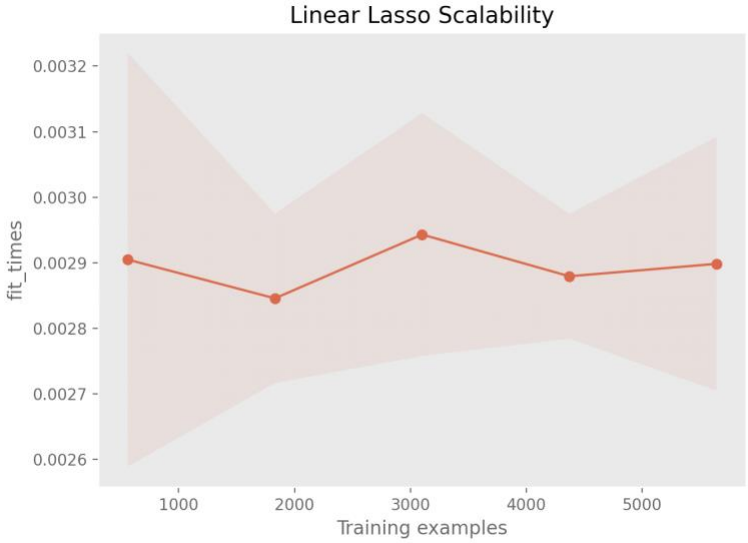

Figure 20. Linear Lasso Scalability

4.3.5 SVM

The SVR (Epsilon-Support Vector Regression) module from scikit-learn allows us to define the kernel function and the gamma which is the kernel coefficient. We evaluate the performance of linear, poly, rbf, and sigmoid as the kernel function, and their mean absolute errors are 0.483, 0.540, 0.439, and 0.614, respectively. Since the rbf function gives the least prediction error, 0.439, we decide to use rbf for further experiments. We also test the model using different gamma values: 0.01, 0.05, 0.1, and 0.5, and find out that 0.05 produces the most accurate prediction with the least mean absolute error, 0.439.

Figure 21 indicates that the training score and cross-validation score are increasing as the number of samples increases. Both scores reach around 0.4 at the end which is close to the result of Linear Lasso. As shown in Figure 22, as the number of samples grows, the fit time is increasing linearly and ending at around 0.95s. The scalability curve looks similar with the regression tree scalability curve since they both start from 0.0 second and end close to 1.0 second. Overall, the learning curve indicates that the prediction accuracy of SVM, 0.40, is at the average compared with other models, and its fit time, in range of 0.0 second to 0.9 second, is also 0.87 second longer than linear models.

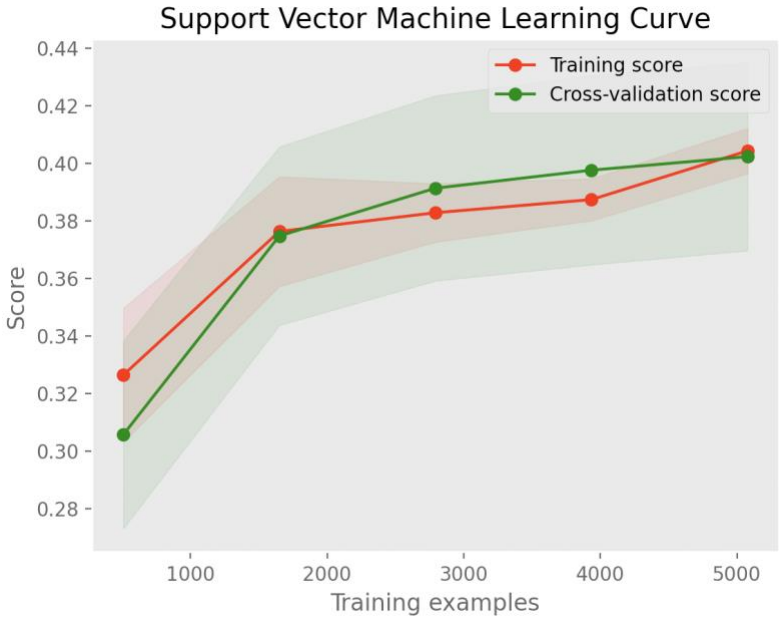

Figure 21. SVM Learning Curve

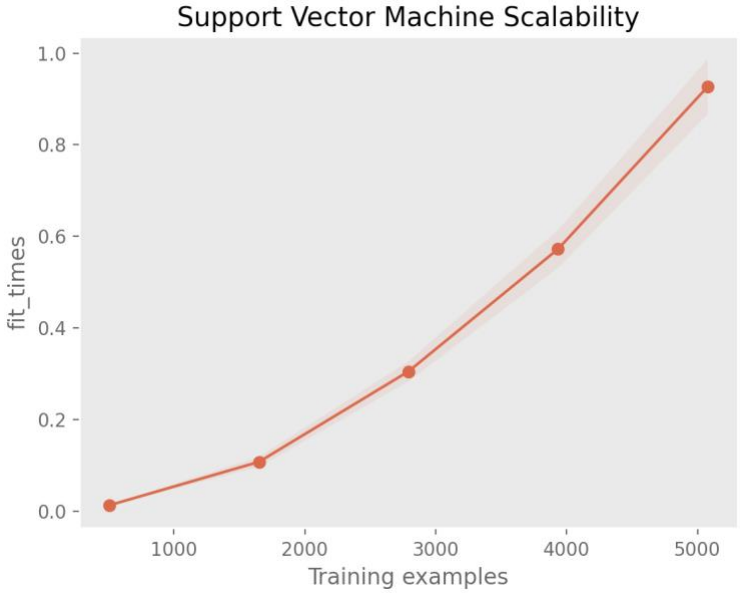

Figure 22. SVM Scalability Curve

4.3.6 Neural Network

The neural network model is implemented using the Sequential model from Keras, which allows us to build a stack of layers that has one input and one output tensor [23]. We build a neural network model with five dense layers and test it with different activation functions. Other parameters such as learning rate, epsilon, decay and number of epochs are also evaluated with different values. Finally, we evaluate the performance of ReLU, tanh, and Sigmoid, and observe that their mean absolute errors are 0.422, 0.627, and 0.501, respectively. We choose ReLU as our activation function since it gives the least mean absolute error, 0.422. The model runs 300 epochs with the learning rate of 0.001 and decay rate of 0.0001.

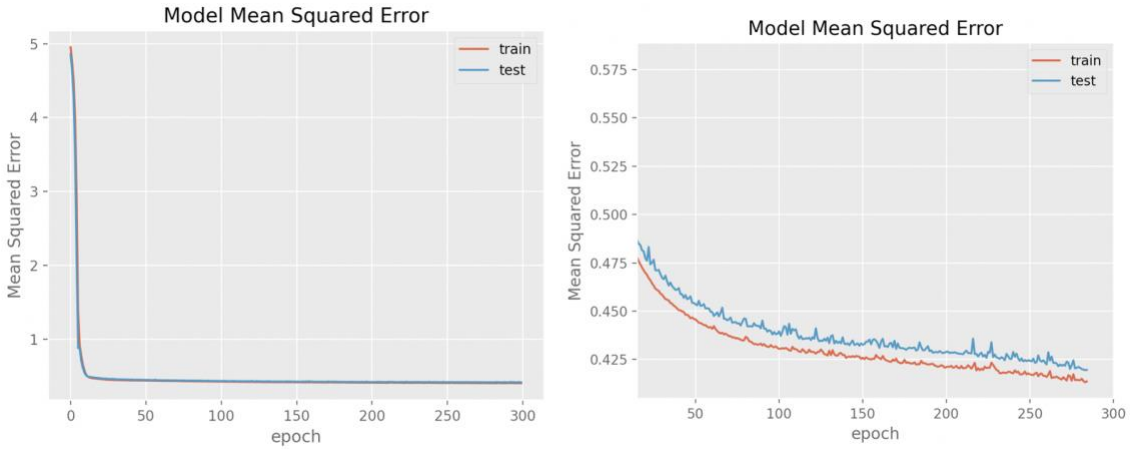

Figure 23. Neural Network MSE

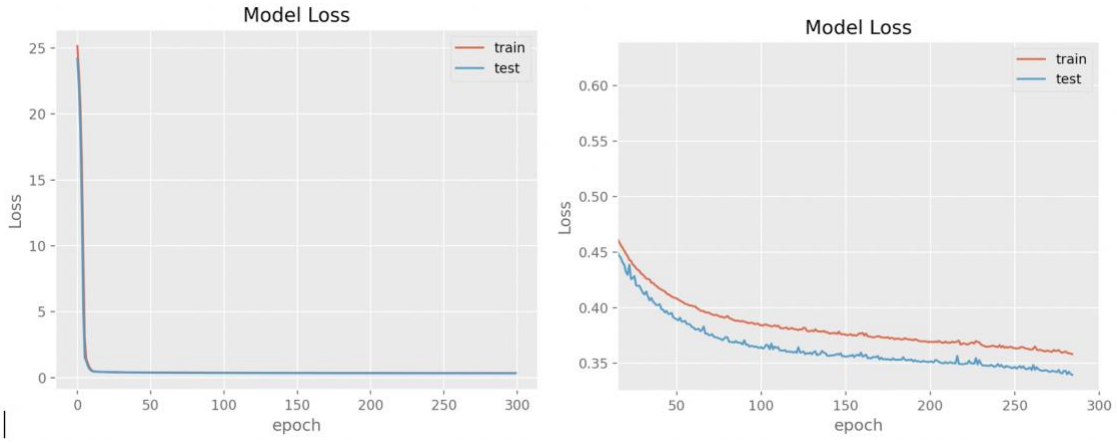

Figure 24. Neural Network Loss Curves

In Figure 23 and Figure 24, graphs on the left side indicate the mean squared error (MSE) and loss during training and validation phases. The MSE and loss drop quickly, which means the model saturates at around the twenties epoch. The two graphs on the right side demonstrate the training status after the saturation point. We find out that MSE and loss continue to drop slowly and end at around 0.40. Overall, the model is highly learnable for the first twenty epochs since the MMS and loss drop fast. Compared with machine learning models, the neural network model takes around 5 seconds longer to finish training, but its prediction accuracy is 9% and 10% better than the Linear Regression and Linear Ridge model, respectively.

4.4 Results

To evaluate the performance of each model, we compare the mean absolute error, median absolute error, mean squared error, and R-Squared of cross validation results between three methods in three tables below. Table 3 shows results of the prediction method without sentiment analysis. Regression Tree with Gradient Boost produce the least mean absolute error, 0.444, and the highest R-Squared, 0.436. The entire performance of this method is the worst compared with the other two methods. For example, the mean squared error for Linear Regression, 0.437, is the highest among three methods. As we consider using the sentiment polarity score as a feature, the prediction errors are reduced, except for Linear Lasso. In Table 4, the mean absolute error for Linear Regression is decreased from 0.463 to 0.455. However, the mean absolute error for Linear Lasso remains at 0.609 and its R-Squared drops from -0.01 to -0.05. Finally, we implement the proposed method, which is to use sentiment classification and subjectivity instead of polarity scores. As shown in Table 5, the performance of all models is improved, except for Linear Lasso. For example, the median absolute error for Regression Tree is decreased from 0.294 to 0.287, but the mean absolute error for Linear Lasso remains 0.609.

	Mean Absolute Error	Median Absolute Error	Mean Squared Error	R-Squared
Linear Regression	0.463	0.355	0.437	0.346
Regression Tree with Gradient Boost	**0.444**	**0.332**	**0.377**	**0.436**
Linear Ridge	0.457	0.350	0.431	0.354
Linear Lasso	0.609	0.501	0.633	-0.01
SVM	0.445	**0.332**	0.422	0.367
Neural Network	0.485	0.371	0.447	0.330

Table 3. Testing Results of Prediction Without Sentiment Analysis

	Mean Absolute Error	Median Absolute Error	Mean Squared Error	R-Squared
Linear Regression	0.455	0.347	0.429	0.358
Regression Tree with Gradient Boost	**0.399**	**0.294**	**0.354**	**0.470**
Linear Ridge	0.457	0.340	0.432	0.351
Linear Lasso	0.609	0.501	0.633	-0.05
SVM	0.439	0.324	0.413	0.380
Neural Network	0.426	0.307	0.389	0.417

Table 4. Testing Results of Prediction Using Sentiment Polarity

	Mean Absolute Error	Median Absolute Error	Mean Squared Error	R-Squared
Linear Regression	0.452	0.334	0.428	0.359
Regression Tree with Gradient Boost	**0.312**	**0.287**	**0.346**	**0.481**
Linear Ridge	0.454	0.339	0.431	0.354
Linear Lasso	0.609	0.501	0.633	-0.05
SVM	0.439	0.326	0.414	0.380
Neural Network	0.422	0.310	0.384	0.425

Table 5. Testing Results of Prediction Using

Sentiment Classification

5. Conclusion and Future Works

In this paper, we introduce machine learning models and a sentiment classification technique to build Airbnb price prediction models. We believe that customer review is a key factor to improve the price prediction since customers and business owners evaluate prices based on previous feedbacks. We point out and implement a proposed prediction method for building the prediction model and evaluate its performance with two prediction methods from previous researches using the public Airbnb dataset for San Francisco. To compare the performance of three methods, we analyze the mean absolute error, median absolute error, mean squared-error, and r-squared value of six machine learning models.

Overall, the proposed method, using sentiment classification and subjectivity as a feature, achieves the least median absolute error, 0.287, with the regression tree model. The prediction method, building the price prediction model without using customer review, produces the highest mean absolute error, 0.609, with the Linear Lasso model. Performances of all selected machine learning models are improved as we consider using sentiment polarity score and classification, respectively, except for the Linear Lasso model. Although we observe overfitting in the Regression Tree with Gradient Boost model, it still achieves the least absolute median error, 0.287, and the highest R-Squared value, 0.481. The Neural Network model perform slightly worse than the regression tree model, which produces the absolute median error, 0.310, and the R-Squared value, 0.425.

For future works, we expect the price prediction model to be improved using a larger dataset with balanced customer reviews since the mean squared error in the Neural Network model is still decreasing at the end of the training phase. Additionally, public Airbnb datasets contain more positive reviews than negative reviews. A well-balanced dataset should be helpful to build a more accurate price prediction model. Other than customer reviews, historical prices might be another key factor to evaluate the price. Customers expect a lower price if the price is constantly decreasing. Lastly, since we only perform experiments using the dataset for San Francisco, we would like to know if the model performs the same using datasets for different cities.

6. References

[1] Luo, Y. et al. “Predicting Airbnb Listing Price Across Different Cities.” (2019).

[2] Tang, E.. “Neighborhood and Price Prediction for San Francisco Airbnb Listings.” (2015).

[3] Nguyen, QuangTrung. “QuangTrungNguyen/Airbnb-Pricing-Prediction.” *GitHub*, github.com/QuangTrungNguyen/Airbnb-pricing-prediction.

[4] Cmdline. “Linear Regression Using Matrix Multiplication in Python Using NumPy.” *Python and R Tips*, 18 Mar. 2020, cmdlinetips.com/2020/03/linear-regression-using-matrix-multiplication-in-python-using-numpy/.

[5] Breiman, L. (June 1997). "Arcing The Edge". *Technical Report 486*. Statistics Department, University of California, Berkeley.

[6] Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (1999). "Boosting Algorithms as Gradient Descent" (PDF). In S.A. Solla and T.K. Leen and K. Müller (ed.). *Advances in Neural Information Processing Systems 12*. MIT Press. pp. 512–518.

[7] Friedman, Jerome. (2000). Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics. 29. 10.1214/aos/1013203451.

[8] Kennedy, Peter (2003). *A Guide to Econometrics* (Fifth ed.). Cambridge: The MIT Press. pp. 205–206. ISBN 0-262-61183-X.

[9] *An Introduction to Support Vector Regression*, towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc 1672c2.

[10] Dinesh Asanka. *Implement Artificial Neural Networks in SQL Server*, https://www.sqlshack.com/implement-artificial-neural-networks-anns-in-sql-server/

[11] Jadon, Shruti. “Introduction to Different Activation Functions for Deep Learning.” *Survey on Activation Functions for Deep Learning*, 3 July 2020, medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092.

[12] Gupta, Shivani & Gupta, Atul. (2019). Dealing with Noise Problem in Machine Learning Data-sets: A Systematic Review. Procedia Computer Science. 161. 466-474. 10.1016/j.procs.2019.11.146.

[13] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang and S. Lin, "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40-51, Jan. 2007, doi: 10.1109/TPAMI.2007.250598.

[14] Andrade, Chittaranjan. “The *P* Value and Statistical Significance: Misunderstandings, Explanations, Challenges, and Alternatives.” *Indian journal of psychological medicine* vol. 41,3 (2019): 210-215. doi:10.4103/IJPSYM.IJPSYM_193_19

[15] Puspita Kencana Sari *et al* 2018 *J. Phys.: Conf. Ser*. **971** 012053

[16] “Simplified Text Processing.” *TextBlob*, textblob.readthedocs.io/en/dev/. Accessed: 2020-09-18.

[17] “Airbnb public dataset.” http://insideairbnb.com/get-the-data.html. Accessed: 2020-10-01.

[18] Raschka, S.; Patterson, J.; Nolet, C. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information* **2020**, *11*, 193. https://doi.org/10.3390/info11040193

[19] J. Menke and T. R. Martinez, "Using permutations instead of student's t distribution for p-values in paired-difference algorithm comparisons," *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, Budapest, Hungary, 2004, pp. 1331-1335 vol.2, doi: 10.1109/IJCNN.2004.1380138.

[20] P. Kalehbasti, L. Nikolenko, and H. Rezaei. 2019. AirBnbPricePrediction. https://github.com/PouyaREZ/AirBnbPricePrediction. Accessed: 2020-09-21

[21] “Sklearn.model_selection.learning_curve.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html#sklearn.model_selection.learning_curve. Accessed: 2021-02-06.

[22] “Sklearn.linear_model.Lasso.” *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html. Accessed: 2021-02-13.

[23] “The Sequential Model.” *Keras*, https://keras.io/guides/sequential_model/. Accessed: 2021-02-17.