Final Assignment 2022-23

Part I

Introduction

In this assignment, I analyzed the data from the Airbnb website to predict multiple continuous rating scores. Two different regression models (Ridge and K-Nearest Neighbors Model (KNN)) will be used to the extracted features to predict the scores. In the last, the importance in each model will be presented to show which feature is most related to the rating scores.

Data Analysis

I downloaded two csv files on the website, "reviews.csv" and "listings.csv".

- **Reviews.csv**

"reviews.csv" has 6 columns and 243183 samples, I only used "listing_id" and "comments" in this table. "listing_id" is used to merge with the "listings.csv". And comments are the users' comments of different listings. I used the text features to analysis the comments.

- For comments, I removed punctuation marks such as the html label <br/> and so on, Then I translated the non-English comments into English with the google API. Google translate API have the access limit for each request. So, I tried multiple times to translate the comments and saved it into the csv file until all the texts are in English. For the emojis, I found that in word2vec, it can be recognized without any process. Then I converted texts to lowercases, removed numbers, and removed whitespaces. I used text pre-processing tools to split input character sequence into tokens, then merged it back to sequence. "Tokenize" split isn't to "is" and "n's" for example. The next step is lemmatization, it grouped together the inflected forms of a word so they can be analyzed as a single item. I grouped all the comments together with the key "listing_id" and concat the comments together for the same listing_id. Finally, I extracted "tf-idf" features to represent the comment and regarded it as the baseline model of this project. I also trained a word2vec within all the sentences in the comments. Then I extracted 300-dimension word2vec features for each word, and averaged features for each word in the same sentence. The result shows that the word2vec feature is better than the tf-idf feature, which I will discuss later.

- **Listings.csv**

"listings.csv" has 75 columns and 7566 samples, "id" is used to align with the comments, so I changed the name to "listing_id" and merged two csv together. I removed all unrelated columns such as the id, date, names, and URLs.

Then I processed all the values with different types into float values.

- For "t" and "f" values, I replace ["t", "f"] into [1.0, 0.0]
  Columns: **'host_is_superhost', 'host_has_profile_pic', 'host_identity_verified'**
- For unique values, I replaced it to indexes of the unique list. For example, the values have "1 bed", "2 beds", "3 beds", I replaced it into [0.0, 1.0, 2.0] separately.
  Columns: **'host_response_time', 'host_neighbourhood', 'property_type', 'room_type', 'bathroom_text'**
- For **"price"** column, it included dollar signs "$", and commas separating sets of three orders

of magnitude in prices. The characters are removed and then convert the string to float value.

- ➢ For columns with "%" character, "%" character was removed, converted the string to float values and divided by 100 to express the rate as decimal.
  Columns: **'host_response_rate', 'host_acceptance_rate'**
- ➢ For float values. I filled the empty values into mean value of the corresponding column.
  Columns: **'host_response_rate', 'host_response_time', 'host_acceptance_rate', 'bedrooms', 'beds'**
- ➢ For sentences, I processed it with the same procedure with the "comments" and got the tf-idf features to represent it.
  Columns: **'neighborhood_overview', 'description'**
- ➢ For float values. I filled the empty values into mean value of the corresponding column.
  Columns: **'host_response_rate', 'host_reponse_time', 'host_acceptance_rate', 'bedrooms', 'beds'**
- ➢ For column **"amenities"**, firstly, I converted all the string values to lists of amenities and converted all the amenities to lowercase. Then I removed all the punctuations and some special labels such as "\u2013" and so on. After that, I merged some similar expressions. For example, if the amenity has the substring "wifi". Then I replace the name into "wifi". Finally, I created an amenities list and count the vector for each row. I also deleted some amenities which appears less than 5 times in all the listings. In the end, 36-dimension features were used to represent the amenities.
- ➢ For all the **review score** components, I removed the rows in which the score value is missing. I think predicting a false value such as the mean value will affect the training and evaluation of the models.

After preprocessing of all the values, I got 4157-dimensional features for each listing and 5058 samples including the float values, (tf-idf features for description, neighborhood overview, comments), bag of word features for amenities. In the meantime, I got 1873-dimensional features for each listing and 5046 samples including the float values, (tf-idf features for description, neighborhood overview), bag of word features for amenities and 300-dimensional word2vec features for comments.
In the last step, I normalized the data for each column with min-max normalization function with numpy to make our data easy to converge and the volume of the data keep the same.

Hyperparameter tunning

For Ridge Regression and K-Nearest Neighbors Model (KNN), hyperparameter tuning was conducted with a 5-fold cross validation. I used the mean squared error (MSE), better performance is smaller value, which means lower on y-axis when plotting.

- **Ridge Regression**

The cost function of Ridge Regression is:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 + \theta^T\theta/C$$

Ridge Regression used the L2 regulation. C is the hyperparameter to control the penalty, and I choose the best hyperparameter with cross validation.
As shown on Figure 1, when C = 0.005, the mean square error keeps the smallest. C = 0.005 was

settled on, the MSE was around 0.11. The C value is very small, and it make our model "simple", that's because our features such as tf-idf are so sparse.

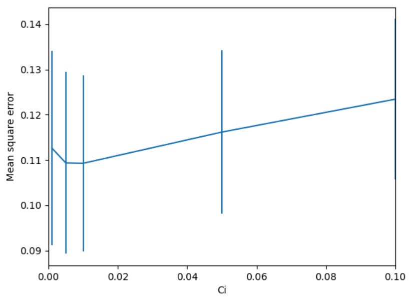


Figure 1 Choose proper C value for Ridge Regression

- **KNN**

The K is the hyperparameter, and I choose the best hyperparameter with cross validation.
`As shown on Figure 2, when K = 100, the mean square error keeps the smallest. K=100 was settled on, the MSE was around 0.12.

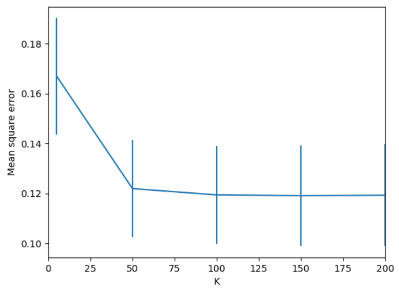


Figure 2 Choose proper K value for Ridge Regression

Model Assessment

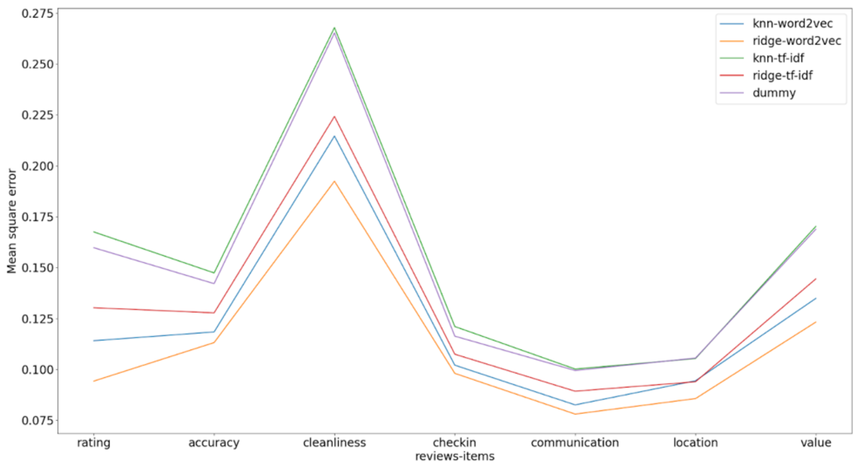


Figure 3 Mean Square Error of different feature and models

In this section, I predicted all the ratings with all the extracted features for each model. I used mean squared error (MSE) and R-squared as the metrics. R-Squared is a statistical measure of fit that indicates how much variation of a dependent variable is explained by the independent variables in a regression model. Bigger value represents the better performance.

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n}(y_i - \bar{y})^2}$$

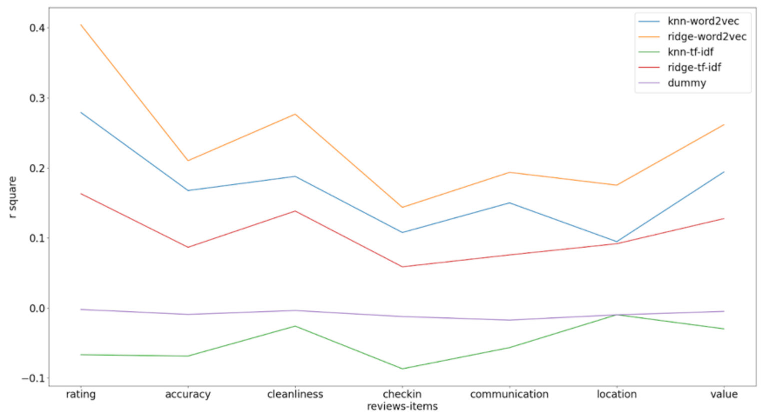


Figure 4 R-squared of different feature and models

I extracted two kinds of features for the comments, word2vec and tf-idf, and concat it with other features. It's obviously that the performance of word2vec was usually better than tf-idf, and the performance of ridge regression model is usually better than KNN with this kind of feature distribution. Except the tf-idf feature with KNN model, our model is usually better than the dummy model which always predict the mean values. It tends out that our model can validly predict all the ratings.

Feature importance

Since Ridge Regression model is linear model, features can be ranked by the coefficient used on the model. I selected the top 20 features for each of the review rating. The feature importance can be seen on Figure 4.

As shown in Figure 4, the red bars represented that the corresponding feature is negatively correlated with the review score which is shown in subplot title, and the blue bars represented that the corresponding feature is positively correlated with the review score.

We can observe that almost all the comments with the word "dirty" decreased the ratings, "host_is_superhost" and the comments with the word "recommend" "great" increased the ratings. Specially, the comments with the word ("dirty" and "clean") influences the review scores of cleanliness most, the comments with the words ("location" "bus" "center" "restaurant", "city", "heart") which are related to the locations influences the review scores of locations most.

By the analysis of the above figures, we can conclude that the feature importance is in line with our intuition. So, I think the models we got is valid to predict the review scores and we can get valued conclusion with the machine learning models.

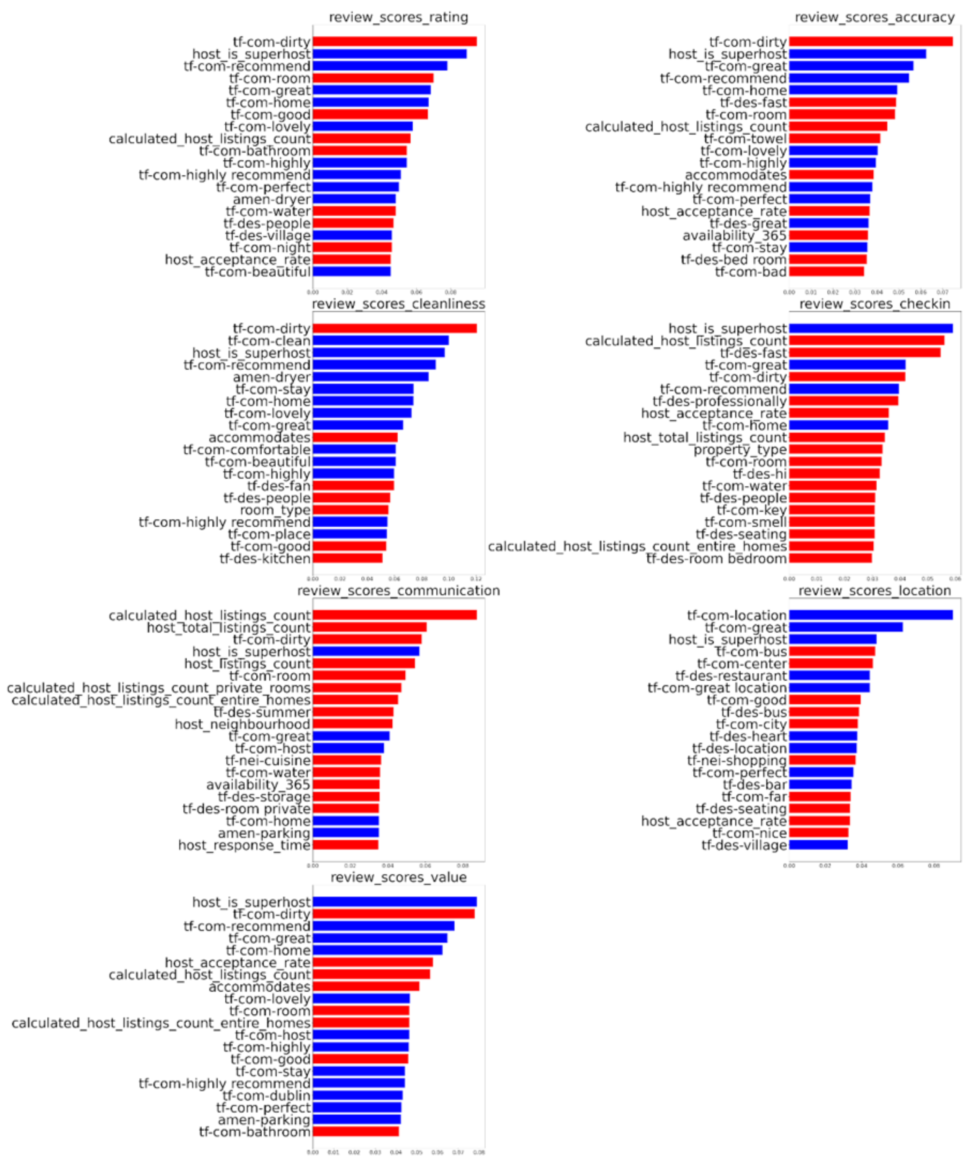
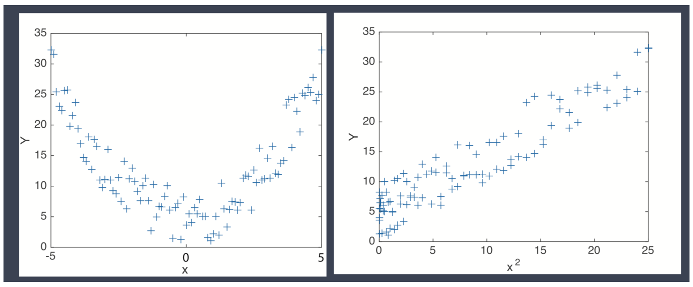


Figure 5 Feature importance

Part II

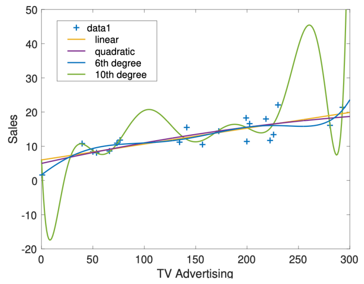
(i)

Example 1: Like the week3 assignment, the data distributes like parabola, if we just use the original x to predict the y, we can just fit a line, and predictions will be inaccurate, we can address this issue with proper feature engineering. In this situation, we can use x^2 as the feature.



Example 2: Overfitting, if the model was too "complex" with less data, the model will fit all the values in the training set. When the model met some unseen data, the prediction will be inaccurate

because the model can only deal with the data in the training set.



(ii) **KNN**

Advantages:

(1) KNN is easy to understand, don't need to estimate the parameters, don't need to train, just save the data distribution.
(2) KNN is not sensitive to outliers. The outliers will be excluded from the K neighbors, and it won't affect the predicting.

Disadvantages:

(1) When the feature has high dimension, every unknown data will calculate the distance with all the known data. It requires a lot of computation, and the prediction will be slow.
(2) When we have unbalanced data, the K neighbors may always belong to the category with more data. It will affect the accuracy of the category with less data.
(3) The classification results cannot be interpreted.

MLP neural net classifier

Advantages:

(1) It can obtain information contained in large amounts of data and build extremely complex models with more hidden layers and neurons.
(2) It can learn non-linear distribution features with the activate function.

Disadvantage:

(1) Complex models need more training time.
(2) The classification results also cannot be interpreted.
(3) The parameter is so much and hard to converge. Like the week 8 assignment, the fc layer has the most parameters.

(iii)

Ideas: By dividing the data and integrating the evaluation results through cross validation, we can "effectively" **reduce the variance in model selection**. In other words, we expect the model to perform well on multiple subsets of the training set rather than on the entire training dataset alone.

The resampling in the cross validation is random. If we use hold-out method, in some situation, most of the "easy classification" samples happens to appear in the test set, it will affect the evaluation of the model. If we don't use the resampling to separate the data into training set and test set, while put all the data to the training set, the performance may be good in the training set and bad in the unseen data, so we cannot evaluate the generalization performance of a machine learning model.

In the meantime, we can get k estimates score with k-fold cross validation, and we can observe that if our model can fit the features validly, if all the k scores are similar, the generalization performance of our model is good.

Choose K value:

- We need to choose proper K value consider the training set size, test set size and the computation. If the training set is too small, the model cannot fit the training set well, it will easily overfitting.
- If the test set is too small, the prediction variance will be so large, considering the test set only has one sample, the validation score will be 0% and 100%.
- If the K value is too large, we need so many calculations.

Considering all the above situations, usually choose the K value as 5 or 10.

(iv)

Lagged out values is a way to use the shift function to shift the value to the next year and use NA to represent the missing data. For example:

Year	Export	Lag1	Lag2	Lag3
1950	1142	NA	NA	NA
1951	1302	1142	NA	NA
1952	2087	1302	1142	NA
1953	1481	2087	1302	1142

For year 1953, we can get the feature which contains the export in 1953 and the last three years. We can train a model like

$$Export\ 1953 = b + a_0 Lag1 + a_1 Lag2 + a_3 Lag3$$

So, we can predict the export of 1953 with the export of last three years. Once we created the lagged-out values, we can shuffle the dataset without influencing the time series data.

Appendix

The code of this project was published on https://github.com/bazingayu/ML_final_assignment

The data is on：

https://drive.google.com/drive/folders/16U4-0n2P1pWx6nR5a00PgazjGOhNgXuE?usp=sharing