Understanding Airbnb Reviews

SIADS 696: Milestone II

May - June 2023

Chi Huen Fong (chfong)

John Kaspers (kaspersj)

Steven David Ong (Steveong)

Introduction

In this sharing economy where online reviews lack anonymity, the act of contributing a negative review amidst a sea of positive ones may amplify psychological discomfort. This stems from an inherent sense of divergence perceived by the author in relation to the opinions of the majority (J.Meijerink, E.Schoenmakers, 2019). From customers' perspective, considering the hosts' autonomy to reject potential guests, it becomes imperative for guests to accumulate favorable reviews as much as the hosts themselves. Thus the stakes are high on both sides as reputation hangs on a balance. Given the unique behavioral idiosyncratic traits observed within the sharing economy, it was discovered that for Airbnb, 95% of properties have an average rating of 4.5 to 5 stars. Virtually no property has 3.5 stars and below as compared to tripadvisor, which has more variance in the reviews and has an average rating of 3.5 stars (Georgios. Z et al., 2018). From a guest perspective, discerning the distinction between ratings such as 4.7 and 4.8 can prove challenging. Alison.P, 2020 mentioned in an article written for the WallStreet Journal ("WSJ") that beyond reliance on the star ratings, consumers should scrutinize the content of reviews as there may be subtle messages unveiled about the quality of the property. Descriptions may allude to potential red flag issues eluding the high rating given.

Findings from past research papers underscore the potential opportunity to investigate questions regarding consumers' reviews, specifically analyzing whether these reviews can provide a more precise and contextualized measure of a guest's unique experience with an Airbnb listing. To achieve this, we will employ a methodology involving the extraction of sentiments from consumer reviews, their subsequent normalization, and classification within the range of 0 to 4 (0 representing the lowest and 4 the highest). Our goal is to construct a machine learning sentiment scoring model capable of predicting and assessing each property's rating based on the text reviews received. This process promises to yield a more nuanced evaluation tool within the sharing economy, enhancing the precision of property appraisals. Once this model is operational, we foresee that hosts would benefit significantly from a more insightful understanding of customer feedback. To achieve this, we propose to implement unsupervised learning in the form of topic modeling on Airbnb reviews. The aim is to uncover prevalent themes within guests' feedback, offering valuable insights into the focal points of guests' concerns. This knowledge will empower Airbnb hosts to address specific areas of concern and effectively meet their guests' expectations, thereby optimizing the user experience on the platform.

Summary of key findings for supervised and unsupervised learning

Finding for supervised learning: Through the supervised learning work, we encountered instances where despite high traditional rating scores, underlying issues were discernible within the review text. Our model successfully identified these negative sentiments, recalibrating the review score based on the text itself to provide a more accurate reflection of the guest experience. This underlines the model's capacity to capture nuanced feedback often overlooked by conventional rating systems, thereby delivering a more comprehensive and authentic appraisal of the property quality.

Finding for unsupervised learning: Through the application of unsupervised learning, we discovered prevalent themes that arise within guest feedback and the analysis exposed key areas of guest concern, shedding light on aspects that significantly impact their experiences. Capability of the model to identify and categorize customer feedback into actionable themes underscores its potential as a vital tool for hosts. It empowers hosts with the insights necessary to address specific areas of concern, allowing them to align their offerings more accurately with guest expectations.

Related Work

The first "Airbnb NYC Price Prediction", can be found on Kaggle.com (https://www.kaggle.com /code /duygut/airbnb -nyc-price-prediction). This study predicts the price a property should be rented out, whereas we are predicting the rating score a property should have based on text reviews received by making sense of key words that make a property stand out from another. The second project similar to ours can be found on Github.com (https://mohamedirfansh .github .io/Airbnb-Data- Science-Project/). The main difference between this project and ours is that the "Airbnb -Data- Science-Project" takes a more exploratory approach to analyzing listings and links review sentiment to price, while our report (and analysis) focuses on the connection between review sentiment and ratings. Also, this project uses different Machine Learning methods to describe price predictors. The third example of an existing Airbnb project is found at https://github.com /susanli2016 /Machine- Learning- with-Python/tree/master, in the the directories: "Airbnb Listing Toronto.ipynb", "Airbnb New User Bookings.ipynb", "Airbnb New User Bookings_2.ipynb". This project's approach utilizes seasonality as a metric for understanding

price, while our project ignores seasonality and we are performing work mainly on review text. Also, this Toronto project dives into Airbnb user information, and sign up methods, which is not part of our work scope.

Data Source

The datasets are open source and it can be downloaded from the "Inside Airbnb" website: http://insideairbnb.com/get-the-data/. We will focus on Los Angeles, California, United States, and use the listings.csv and reviews.csv to explore our analysis.

Dataset	Rows vs Cols	Time period	Description
Reviews.csv (407mb)	1398664, 6	Mar.22 to Mar.23	This dataset encompasses user comments on rental units following their usage. It comprises essential information such as the listing ID, reviewer's ID and name, date of the review, and the accompanying comments provided by the users.
Listings.csv (117mb)	42451, 75	Mar.22 to Mar.23	Each row represents a unique transaction within the Airbnb platform. These transactions encompass a comprehensive range of information, including details about the rental unit, such as its price, location, and number of bedrooms, as well as review score.

Note: We have also created and shared a sample file containing first 100 records for easy reproduction of our data frame.

Initial preprocessing: In reviews.csv, we removed all missing values (327 lines) since the missing values constituted a relatively small portion compared to the total dataset size of over 1 million data. Furthermore, we removed unnecessary columns from the listings.csv file, retaining key columns that are important for our project objectives. To handle missing values in listings.csv, we filled the NaN values in different columns with specific values as outlined in below table.

Columns	Process
host_response_rate, host_acceptance_rate, beds, bathrooms_text and bedrooms, and 7 scores (rating, accuracy, cleanliness, checkin, location, communication and value)	Fill in the NaN by 0
host_response_time	Fill in the NaN by ‘no response’
host_is_superhost, host_has_profile_pic and host_identity_verified	Fill in the NaN by ‘f’

Feature engineering

Data pre-processing steps for reviews dataset: In our analysis, we initially employed a test subset of 10,000 rows from the reviews dataset, allowing us to conserve time and resources and limit error propagation. We utilized the “langdetect” library to identify comment languages, assigning an "Error" label where detection proved unreliable. We identified several issues including emojis, HTML tags, multilingual comments, and punctuation-only entries. Our cleaning strategy involved removing punctuation-only comments and HTML tags using regular expressions, and converting and subsequently eliminating emojis with the emoji.demojize function. Following the successful application of this process on the subset, we extended it to our full dataset, also discarding entries labeled "Error" due to the presence of special characters or empty strings. We found over 90% of the comments in our dataset were labeled as English. As we had more than 1 million rows of data and the process of translating non-English languages to English will be time consuming, we removed comments that were non-English. We also removed comments that consisted of fewer than 3 letters. This involved eliminating entries that contained single-letter or two-letter comments, such as "C" or "o".

Data pre-processing steps for listings dataset: We transformed various columns from text to numeric values. These include 'host_is_superhost', 'host_has_profile_pic', 'host_identity_verified', 'host_response_time' and 'instant_bookable'. For 'price', 'host_acceptance_rate', and 'host_response_rate', we converted strings into numeric

percentages. We standardized 'bathrooms_text' into two new columns, 'bathroom_type' and 'bathroom', using regex and replace functions. For 'amenities', we created 10 columns for the top amenities, assigning 1 if present, 0 if not. Finally, we used get_dummies to convert 'room_type' into indicator variables.

SUPERVISED LEARNING

Objective and motivation

The motivation for this supervised learning arises from several factors linked to the unique dynamics of the sharing economy and the particular role of reviews in platforms like Airbnb. Existing literature mentioned in the introduction of our project report suggests that consumers might face certain mental discomfort when leaving negative reviews due to the non-anonymized nature of such platforms. Consequently, a significant majority of properties on Airbnb have received high average ratings, making it difficult for consumers to discern the difference in quality or experience between listings. Furthermore, as reviews play a vital role in shaping the reputations of both hosts and guests, a skewed or non-representative review system could potentially impact the equitable functioning of the platform.

Key issue noted in the existing review system is that subtle messages about the quality of the property could be lost amidst high overall ratings. For eg. An actual review of a property with a commendable rating score of 4.75 /5.0 received the following review - *"Very nice and quiet neighborhood. The house was clean, but no supplies whatsoever. Be prepared to buy basic necessities like toilet paper, dish wash soap, and even a dust pan."* (Refer to our error analysis segment for more insights). This example underscores the need for a more granular / subtle evaluation method to capture nuances. Proposed sentiment scoring model using review text aims to fill this gap with a comprehensive, detailed and equitable system of evaluation that benefits all users of the Airbnb platform.

Learning methods

We employed a variety of modeling techniques to encompass different types of algorithms and data. The models represent a good mix of methods: linear, probabilistic, tree-based, and transformer-based models for NLP. We included models-Linear Regression, Random Forest, Naive Bayes, Linear Support Vector Machine and transformer-based models - Bidirectional Encoder Representations (BERT) and DistilBERT -a streamlined version of the BERT model, which adds to the diversity of our model selection.

Labels - Bad, Nay, Average, Good, Awesome!

We employed Valence Aware Dictionary for Sentiment Reasoning (VADER) model to perform sentiment analysis on text reviews. This lexicon-based model effectively gauges the polarity (i.e., Positive, Negative) and intensity of emotions expressed in each text review. The VADER model, incorporated from the NLTK package, generates sentiment scores by mapping lexical features to emotion intensities. These sentiment scores were utilized as an alternative to the conventional star rating system.

Reason for classification approach: Choice to use a multiclass classification approach was based on a few considerations. Objective is to predict the rating of the accommodation, which is a categorical variable. The label system, divided at 0.25 intervals, was thought of from the user perspective based on the articles we have cited in the opening introduction. Goal is to predict the rating of the accommodation based on categorical variables with 5 different levels. This reflects the user experience more accurately, as users do not express their sentiments on a continuous scale, but they categorize their experience into distinct levels ('bad', 'awesome', 'good', 'average', 'bad').

Classification models offer interpretability that could be beneficial for our task. They allow us to better understand which categories (or sentiment ranges) the model struggles with and identify potential areas of improvement which we have identified when we used the various supervised learning approaches - 1. Naive Bayes, 2. Logistics Regression, 3. Random Forest, 4. Linear Support Vector Machine ("traditional machine learning") methods.

How VADER was deployed: VADER was applied directly to the raw comments column in our dataframe, providing us a comprehensive capture of sentiments and emotional intensity. VADER model generates a dictionary of four keys: 'neg', 'neu', 'pos - sentiment', and 'compound' - polarity score. 'Neg', 'neu', and 'pos' represent negative, neutral, and positive sentiments respectively, while the 'compound' key signifies the cumulative polarity score of each word in the lexicon, illustrating the overall sentiment intensity. The compound polarity score can range from -1, which implies extreme negative sentiment, to +1, reflecting extreme positive sentiment.

A normalization function was implemented to transform the sentiment scores to a 0-4 scale, represented as {"Bad": 0, "Nay": 1, "Average": 2, "Good": 3, "Awesome!!": 4}. After the tagging process was completed, the statistics of the label column showed a class imbalance where there are much fewer reviews with labels - 0 and 1 as compared to labels - 2, 3 and 4 (Figure 1) :

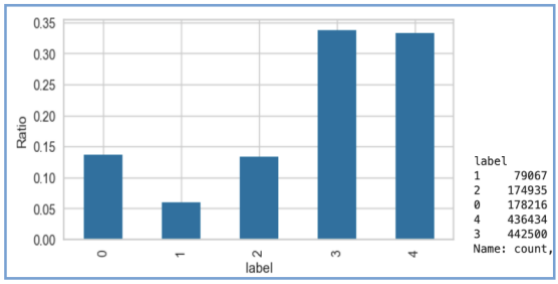


Figure1: Imbalanced data

In such cases, the machine learning models will be biased towards the majority class - 3 and 4, causing poor prediction performance on 0, 1 and 2 as most machine learning algorithms are designed to maximize overall accuracy, which can be achieved by largely focusing on the majority class.

To address this issue, undersampling was done to balance the data (Figure 2). The technique works by randomly eliminating instances from the majority class to prevent its large number from skewing our model. The benefits of undersampling is as follows:

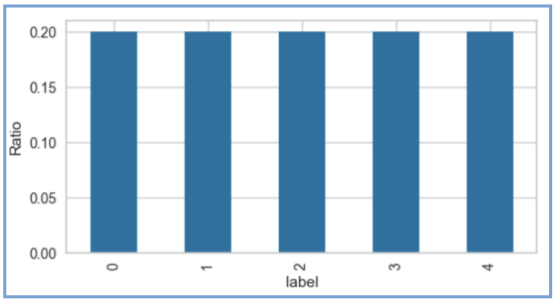


Figure2: Balanced dataset after undersampling performed.

1. *Improves model performance on the minority class:* By undersampling the majority class, we can make the dataset balanced which allows the model to have a better performance on the minority class. It can help to increase the sensitivity/recall of the model.
2. *Reduces the computation time:* By reducing the number of instances from the majority class, we reduce the total number of instances that the model needs to learn from. This can make the training process faster.
3. *Helps to avoid overfitting*: When a model is trained on imbalanced data, it might overfit to the majority class and perform poorly on minority class. By undersampling, we can avoid this overfitting issue.

Feature representation

Transitioning our machine learning models from raw text data to a numerical representation was done via Term Frequency-Inverse Document Frequency (TF-IDF). This particular measure, a quantitative reflection of word significance, forms the basis for our word vector extraction from the TF-IDF vectorizer. The rationale behind this technique is its potential to provide an authentic representation of overall sentiment polarity, attributing weightage to terms based on their distinctiveness and relevance, rather than sheer frequency. TF-IDF is a statistical measure with its own limitations - primarily, its disregard for word order within a sentence. To mitigate this concern, transformer models

```
def extract_features(df_train, df_dev, df_test):


    ### vectorization
    vectorizer = TfidfVectorizer(analyzer='word',
                                 stop_words='english',
                                 ngram_range=(1, 2),
                                 lowercase=True,
                                 min_df=5,
                                 binary=False)
    X_train = vectorizer.fit_transform(df_train["comments2"])
    X_dev = vectorizer.transform(df_dev["comments2"])
    X_test = vectorizer.transform(df_test["comments2"])
    y_train = df_train["label"].tolist()
    y_dev = df_dev["label"].tolist()
    y_test = df_test["label"].tolist()

    return X_train, X_dev, X_test, y_train, y_dev, y_test
```

Figure3: Transforming raw text data to numerical representation.

BERT and its distilled variant Distilled BERT were incorporated within the deep learning section of our study, enhancing our text analysis capability. As shown in Figure 3, vectorizers from scikit-learn with the following key settings was used: (1) Using unigram and bigram, (2) Setting minimum document frequency as 5, (3) Converting all characters to lowercase letters, (4) Removing English stop words. Transformer based models (eg. BERT) was applied on contextual data without going through the vectorizers. Further description will be provided in the transformer-based section.

With our data comprehensively undersampled, it was segmented into three distinct subsets - Training, Validation, and Testing. Random sampling was used to ensure impartial distribution and adhere to a standard ratio of 80%, 10%, and 10% respectively. To prevent data leakage, the TF-IDF vectorizer was independently applied to each data set. To ensure the minimum frequency of each token, a lower limit of 5 words through the min_df parameter in the TF-IDF vectorizer. This ensures a base level of repetition for every term included in analysis, reinforcing the reliability of our feature set.

Multiclass Classification Model

In our exploration of suitable models for property recommendation based on guest reviews, four traditional machine learning methods: Linear Regression, Naive Bayes, Logistic Regression, and Random Forest were employed. Primary aim of our analysis was to select a model that maximizes the F1 score. The F1 score reflects the balance between precision and recall, hence is an appropriate measure for our task given the importance of both false negatives and false positives in property recommendations. Choice of F1 score will be elaborated in our hyper parameter tuning process.

In all four learning methods, these models were trained on the feature set derived from the guest reviews (cleaned_comments) using the fit_transform method. This ensured that the text data was converted into a numerical representation that could be processed by the machine learning algorithms. Following this, each model's performance was assessed by comparing the validation target labels to the predictions generated by the model on the validation set. To ensure robustness and avoid overfitting, 10-fold Cross-Validation (CV) was used during the model training phase. By dividing the dataset into 10 parts, training the model on 9 parts, testing it on the remaining part, repeating this process ten times, the results provided a more reliable and generalizable performance metric. End results is an unbiased model evaluation metrics that provides insight into how well the model would generalize to unseen data.

	accuracy	precision_weighted	recall_weighted	f1_weighted	roc_auc_ovr
Logistic Regression	0.554	0.549	0.554	0.551	0.848
Naive Bayes	0.415	0.423	0.415	0.406	0.716
Random Forest Classifier	0.556	0.552	0.556	0.540	0.842
Linear Support Vector Machine (SVM)	0.316	0.409	0.316	0.258	NaN

Figure4: Metrics used for classification prediction model evaluation.

Evaluation of results

The traditional machine learning model's accuracy score was relatively lower than expected. We had initially hypothesized five distinct sentiment labels, corresponding to a spectrum of "Bad" (0), "Nay" (1), "Average" (2), "Good" (3), and "Awesome!!" (4) for comprehensive sentiment classification. It was clear during evaluation that distinction between sentiments - "Average", "Nay", and "Good" was a challenging task for the models. We provide a classification report (Figure 5) and confusion matrix (Figure 6) for our highest-performing model, the Logistic Regression Model, prior to hyperparameter tuning. The intricate and nuanced distinctions between sentiment labels seem to blur, resulting in a substantial overlap and a narrow margin for error. Consequently, this led to an upsurge in misclassification rates, which subsequently reduced our overall accuracy scores. This observation is consistent with the performance of other traditional Machine Learning models. For a similar performance report on Naive Bayes, SVM, and Random Forest Classifier, please refer to Appendix 2A.

Logistic Regression

Testing Set
Accuracy: 0.5591642636717762

	precision	recall	f1-score	support
0	0.5805	0.6098	0.5948	7943
1	0.4741	0.4887	0.4813	7954
2	0.4557	0.4122	0.4328	7878
3	0.4965	0.4542	0.4744	7908
4	0.7572	0.8325	0.7931	7851
accuracy			0.5592	39534
macro avg	0.5528	0.5595	0.5553	39534
weighted avg	0.5525	0.5592	0.5550	39534

Figure5: Logistic regression model classification report

Figure6: Logistic regression model confusion matrix

While the results presented challenges, this exercise was valuable, highly informative in understanding the performance boundaries of traditional learning models when it comes to discerning between subtle sentiment variations. We anticipate intriguing insights into how transformer-based models like BERT and DistilBERT handle these nuances. The exploration of these models and their misclassification rates will be the focus of the upcoming Transformer Model section in our paper.

Hyperparameter Tuning Description and Results

Hyperparameter tuning was performed on the logistic regression model since it has the highest f1-score amongst the four traditional machine learning models (Figure 4). Default logistic regression model was fitted on the training and the defined search space for RandomisedSearchCV instance (Figure 7) is as follows:

```
search_parameters = {"solver": ['lbfgs', 'liblinear'],
        "penalty": ['none', 'l1', 'l2', 'elasticnet'],
        "C": [0.001, 0.01, 0.1, 1, 5, 10],
        "multi_class": ['auto', 'ovr']}

search = RandomizedSearchCV(clf_lr, search_parameters,
                            n_iter = 500,
                            scoring = 'f1_macro',
                            random_state = 42)

result = search.fit(X_train, y_train)
result.best_params_
```

Figure7: Defined search space for RandomisedSearchCV instance

Our choice of F1_macro as the evaluation metric is motivated by a few important considerations. (1) The F1_macro score provides a balanced measure of the model's performance in terms of both precision and recall. Precision assesses how many of the predicted positive instances are truly positive, while recall evaluates how many actual positive instances are correctly identified by the model. (2) The F1_macro score is suitable for multi-class or multi-label classification tasks, such as ours, that may involve imbalanced classes. The "_macro" refers to the calculation method where the metric is calculated separately for each label and the results are averaged. This ensures that all classes, regardless of size, are accounted for in the final metric.

In the context of Airbnb reviews, using F1_macro effectively gauges the model's performance across various sentiment classes, from 'Bad' to 'Awesome!!'. It ensures that the model's capability to correctly classify less frequent sentiments (which might be 'Bad' or 'Awesome!!' reviews) is as important as its capability to classify the more common sentiments. Hence, this makes F1_macro a particularly suitable metric for our research objectives.

The best parameters after executing the search results are as follows: {'solver': 'lbfgs', 'penalty': 'l2', 'multi_class': 'auto', 'C': 5}. With the best parameters incorporated with the logistic regression model, we noted the tuned model performed better in all performance metrics presented in Table 1 as follows -

Evaluation of performance between tuned and default logistic regression model

Classifier	Accuracy (Balanced)	Precision (Macro)	Recall (Macro)	F1 Score (Macro)	AUC ROC Score	Log Loss
LogisticRegression (Default)	0.578	0.572	0.579	0.575	0.861	0.995
LogisticRegression (Tuned)	0.581	0.579	0.582	0.580	0.863	0.983
Difference (%)	*0.52%*	*1.19%*	*0.52%*	*0.97%*	*0.15%*	*-1.18%*

Table 1: Summary of results for tuned and untuned logistic regression model.

The Log Loss is lower for the tuned logistic regression when compared to the default model, this can be interpreted as the tuned model is performing better. This is because the tuned model's predicted probabilities are closer to the actual class labels in the dataset, implying a more accurate classification. In other words, the probabilities predicted by the tuned model are more accurate, which means that the model has a higher level of certainty about its predictions.

As shown in the facing chart (Figure 8), ROC was above 0.5 for all classes and the curve comes closer to the top left corner of the plot. The model looks promising in distinguishing between the different classes.

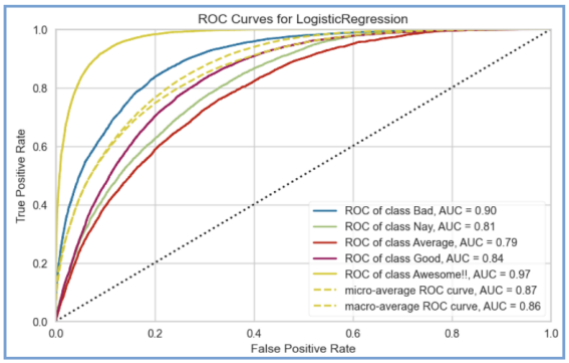


Figure8: ROC curve for logistic regression model

Ablation and feature importance analysis

From the learning curve study as shown in Figure 9, it is evident that the size of the training set plays a crucial role in the performance of our models. As training set size increases, we observe an improvement in the cross-validation score, indicating enhanced generalization performance on unseen data. This suggests that larger volumes of data provide the model with a broader range of information to learn from, thereby increasing its ability to predict new inputs accurately.

Simultaneously, we observe that the training score decreases with increasing training set size, suggesting a reduction in overfitting. In smaller training sets, models can memorize details and noise, leading to high accuracy on the training set but poor generalization to new data. As training size increases, our model becomes more robust, less susceptible to noise, resulting in a lower training score but enhanced ability to generalize. Analysis indicates that expanding the training size has contributed positively to our prediction success, striking a balance between learning from data and maintaining a model complexity that allows effective generalization.

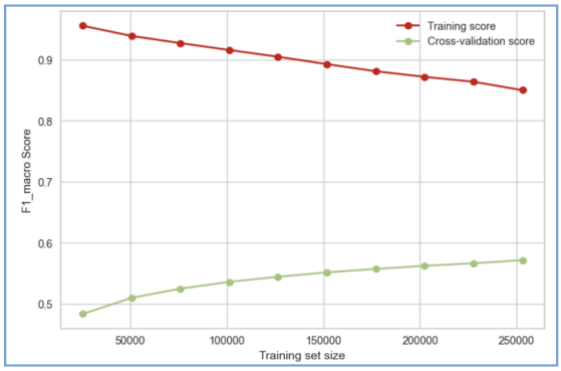


Figure9: Learning curve

Error Analysis

We analyzed errors in predictions our model has made in order to gain insights on how we can refine the model. For those line items where there are errors in predictions when compared to the labels, we looked into the comments and sentence structure and selected 5 samples for analysis (Table 2) as follows. For full details, refer to the error analysis performed in the T5_Supervised_Learning Notebook.

Logistics regression model error analysis

No.	Line item	Rating (1)	True Labels	Predicted Labels	Review Text
1	153490	4.75	Nay	Bad	In reading the past review I agree with most everything. We were there to attend wedding, so we did not spend much time there. The house needs attention to detail. Nice but could be spectacular with a few improvements.
2	832706	4.75	Nay	Bad	Very nice and quiet neighborhood. The house was clean, but no supplies whatsoever. Be prepared to buy necessities like toilet paper, dish wash soap, and even a dustpan.
3	1005236	4.75	Nay	Bad	A bit overvalued for a place without air conditioning. But comfortable and centrally located.
4	616153	4.75	Nay	Bad	Communication with Rosa was easy, and she responded very quickly. She was also very warm once we arrived and showed us the beautiful view from the balcony which is truly beautiful at night. However, the space was not clean. The furniture in the room was extremely dusty and the bathroom was quite dirty especially the bathroom mirrors and toilet seat. I also didnt feel the linens and covers were fresh. My stay was okay but i was not happy with the less than clean space.
5	1110608	4.75	Average	Bad	We loved being able to have a golf cart with our Airbnb reservation. The condo is in a great location with private beach. I was however disappointed on how sorry it was. You can tell it is not properly cleaned. We saw several spiders. Spider webs on the side of the bed dust etc. Theres no AC it was a bit musty. Besides that, it is a cute place that needs some TLC.

Bad review

High score

Note (1): Rating was based off the rating given to a property by the end consumer

Table 2: Five samples of classification errors noted.

In our analysis of the predictive capability of our Logistic Regression model on customer reviews, we observed certain nuances in the data that may have contributed to the model's misclassifications. A particular pattern stood out in several instances where the reviews initiated with a positive comment, only to be followed by negative remarks. Such mixed sentiments present a considerable challenge for the model's accurate prediction due to their inherent contradiction and ambiguity. Three main findings, reasons for wrong prediction were notable from our error analysis:

Finding 1: The technique employed to transform the review text into numerical features was Term Frequency-Inverse Frequency (TF-IDF) Vectorization. Although it's a widely utilized method in text data processing, it exhibits a notable limitation. Specifically, it treats each word in the text as an independent entity, neglecting the sequential nature and contextual semantic relationships between words. This simplification could contribute to a misinterpretation of the sentiment, thereby causing the model to misclassify the review.

Finding 2: Our choice of model, Logistic Regression, despite being a robust and effective tool for various classification tasks, may not be optimally suited for the intricacies of textual data. It's a linear model, and the complexity and multidimensional associations in text data might be too convoluted for such a model to fully comprehend. More sophisticated models, such as Bidirectional Encoder Representations from Transformers (BERT)), might prove more adept at deciphering these complex patterns and yield improved results.

Finding 3: Upon analyzing the errors in our model, we noted that there may be an inherent ambiguity in distinguishing between labels 1 and 2. These categories might represent sentiments that are subtly different and thus harder to demarcate clearly. This difficulty is exacerbated by the subjective nature of sentiment interpretation, where one individual's slight annoyance could be another's neutral observation. The existing model, as effective as it is for clear-cut categorizations, may be encountering challenges with such finely nuanced sentiment distinctions.

However, our current Logistic Regression model has still demonstrated value by correctly identifying negative sentiment in several reviews. For instance, in the case of review 616153, while the review was associated with a relatively high rating of 4.75, the text within the review hinted at certain issues with the property. The model was able to pick up these signals and classify the reviews accordingly, illustrating the potential of our approach, even amidst some areas of misclassification.

Our initial ambitions, which included the prediction of five distinct categories, may have been somewhat high reaching, leading to misclassification between closely aligned classes such as 'Nay', 'Average', and 'Good'. It seems that these overlapping sentiment domains introduced a certain level of complexity that resulted in prediction inaccuracies. A potential path for future refinement could involve simplifying our classification system to three broader categories - 'Bad', 'Neutral', and 'Good'. This consolidation may provide a more forgiving margin of error, thereby enhancing the model's performance and prediction accuracy. The above results from traditional machine learning models were highly informative in understanding the performance boundaries of our traditional learning model when it comes to discerning between subtle sentiment variations. Learning from the above, it raised our curiosity on whether transformer-based models - BERT (Bidirectional Encoder Representations from Transformers) will perform better with sentiment variation nuances. By pitting BERT against traditional models, we aim to unravel any further nuances and complexities within the data that could be leveraged with BERT to improve our predictions.

Transformer Based Model

In recent years, transformer-based models became important Natural Language Processing (NLP) landscapes and have continued to outperform many other state-of-the-art methods (Karl H. et al, 2020). Two transformer-based models - BERT, Distilled-BERT were used on our Air-BnB review text rating prediction task as we wanted to evaluate how good BERT NLP models benchmark against traditional machine learning models.

- BERT: First applied transformer-based model was BERT. BERT (Bidirectional Encoder Representations) is a language representation model, where deep bidirectional representation from unlabeled text is pre-trained on both left and right context for masked language model and next sentence prediction tasks. Such a pre-trained BERT model can be used for wide tasks after being fine-tuned with just one additional output layer.

- Distilled BERT: Second applied transformer-based model is Distilled BERT. Distilled BERT is a distilled version of BERT, where 97% language understanding capabilities of BERT are retained with 60% faster performance and 40% smaller model size. The Classification report of the BERT and distilled BERT models as follows - Figure 10 with corresponding confusion matrix report in Figure 11:

BERT

	precision	recall	f1-score	support
0	0.975	0.967	0.971	7943
1	0.941	0.941	0.941	7954
2	0.951	0.907	0.928	7878
3	0.926	0.902	0.914	7908
4	0.919	0.994	0.955	7851
accuracy			0.942	39534
macro avg	0.943	0.942	0.942	39534
weighted avg	0.943	0.942	0.942	39534

Distilled BERT

	precision	recall	f1-score	support
0	0.978	0.960	0.969	7943
1	0.944	0.927	0.936	7954
2	0.930	0.942	0.936	7878
3	0.950	0.861	0.904	7908
4	0.890	0.995	0.940	7851
accuracy			0.937	39534
macro avg	0.938	0.937	0.937	39534
weighted avg	0.939	0.937	0.937	39534

Figure10: BERT and Distilled-BERT model classification report.

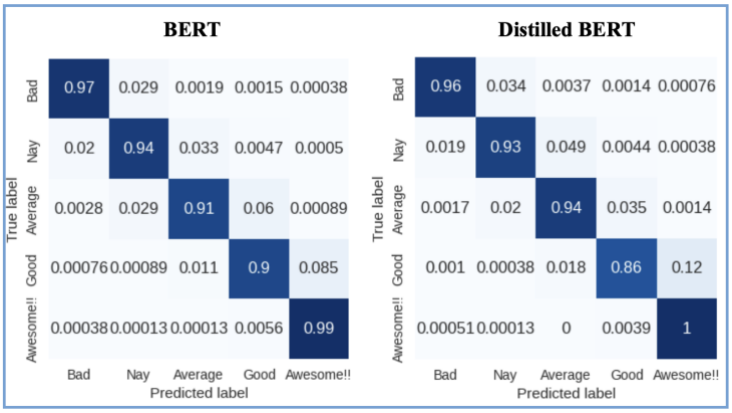


Figure11: BERT and Distilled-BERT model confusion matrix.

```
### testing the model
new_sentence = "This is a great residence, value for money. Tops!"
tokenized_sentence = tokenizer.encode(new_sentence, add_special_tokens=True)
input_ids = torch.tensor([tokenized_sentence]).to(device)
with torch.no_grad():
    outputs = distilled_model(input_ids)

predictions = torch.nn.functional.softmax(outputs[0], dim=-1)
predicted_class = torch.argmax(predictions).item()

print(predicted_class)
```

```
3
```

Figure11a: Testing of prediction capability using Distilled BERT model.

Contrasting traditional machine learning techniques, the performance outcomes of BERT and Distilled-BERT significantly exceeded expectations. F1_Macro and accuracy scores at 0.942 and 0.937 (Figure 10) notably outstripped those of conventional machine learning models. On comparing Distilled BERT with BERT, the two models were on par in terms of accuracy and F1_Macro scores, yet the Distilled BERT model demonstrated superior efficiency, training in half the time needed for the BERT model. As demonstrated in Figure 11a, we tested the distilled BERT model with new data and the model demonstrated strong performance, underscoring its efficacy in dealing with new data. In conclusion, BERT and Distilled-BERT demonstrated a superior performance in handling these nuanced sentiment categorizations, thereby yielding better results.

This signifies the promise of deep learning models in effectively capturing the intricate nature of human sentiments, which may be a valuable direction for future research.

Trade Off Analysis / Conclusion

It is apparent from the above exercise that transformer models notably outperformed traditional machine learning models when evaluated using accuracy and F1_score metrics (Figure12). However, the considerable resource demand of these transformer models, specifically BERT, posed a substantial challenge. The BERT and Distilled BERT model necessitated about 5.5 hours of training time and 40 GPU units, resulting in a time and resource-intensive process. Despite the superior results of the transformer models, we found that the logistic regression model provided a competent and more resource-friendly alternative for our supervised learning task. Balancing performance, cost, and time-efficiency is a crucial aspect of model selection and deployment and we feel logistic regression model is sufficient to meet requirements of our supervised learning task.

Key metrics vs time taken

Machine Learning Model	Accuracy	Precision	Recall	F1 Score	Time Taken
Traditional models (10 Fold)					
Logistic Regression	0.554	0.549	0.554	0.551	10 min
Naive Bayes	0.415	0.423	0.415	0.406	10 min
Random Forest Classifier	0.556	0.552	0.556	0.540	10 min
Linear Support Vector Machine	0.316	0.409	0.316	0.258	10 min
Transformer models					
BERT	0.942	0.943	0.942	0.942	5 hrs 23 min
Distilled BERT	0.937	0.939	0.937	0.937	2 hrs 30 min

Figure12: Key metrics vs time taken.

Discussion

Key learning we gathered from the supervised learning work was learning curve analysis which indicates that expanding the training size contributes positively to our prediction success, striking a balance between learning from the data and maintaining a model complexity that allows for effective generalization. This highlights the importance of having sufficiently large training data in machine learning tasks, serving as a valuable insight for future work and considerations. What surprised us is the much lower accuracy and F1 score for traditional models when compared to the score obtained from distilled BERT and BERT models.

One of the primary challenges we faced during the project was the consistent timing out of our initial BERT model function. This proved to be a significant roadblock that required considerable time and effort to surmount. We repeatedly reached the memory limit on Google Colab Pro, despite its advanced resources. This issue pushed us to consider training the model on a smaller dataset. However, we understood the implications that could have on the validity and generalizability of our results, as the reduced dataset might not accurately represent the wider context. We succeeded in enhancing the efficiency of our codes. We restructured the BERT model training process into smaller, more manageable segments, without compromising the extent of the training data. We are proud of our achievement in developing a model capable of processing a substantial dataset of 300,000 entries. This experience demonstrated the importance of tenacity, efficient coding, and methodical problem-solving in data science projects.

Training a BERT model has been a computationally intensive and time-consuming task. If provided with additional time and GPU resources, we would aim to expand the research on the DistilBERT model by exploring its performance when trained on varying amounts of data, such as half or even a quarter of the current training dataset. This experiment could offer insights into how sensitive the model is to the volume of training data and provide indications on the optimal amount required for desirable results.

We are also curious about the impact of tokenization length on the model's performance. The current model operates with a maximum token length of 256. I propose altering this limit to 128 tokens per sequence. Such an adjustment may affect the model's ability to understand and process information, offering valuable insights into the balance between resource allocation and performance in NLP tasks.

UNSUPERVISED LEARNING

Objective and motivation

If Airbnb ratings are to be based on the text reviews which give new revised scores, unit owners renting out their properties will need to know based on the new scoring model, what are the words that may set them apart from a nay or a yay in topic modeling. Because most of the review scores in Airbnb are positive, meaning over 4.5, we thought it made sense to dive into the topics within the text reviews. In terms of a workflow, we first examined the common words in reviews using WordClouds (see appendix 3A). After splitting the data based on Vader sentiment into positive, negative, and neutral (Refer to Supervised Learning - Labels - Bad, Nay, Average, Good, Awesome! section), we noticed that many of the words are the same. This led us to believe that there exists a great amount of commonality in phrases across reviews, regardless of sentiment. Because of this, we wanted to find common topics to understand similarities and differences between the textual reviews. We did this to understand what makes certain reviews stand out compared with others. We used LDA (probabilistic) and NMF (non-probabilistic) for topic modeling on text reviews to extract topics from text data. By applying these methods to the text reviews, we aim to find the main topics guests are talking about in their reviews, providing the host valuable insights about what customers care about.

LDA

Latent Dirichlet Allocation (LDA) was the probabilistic chosen method used to extract hidden thematic structure from our large collections of review text. Unlike other clustering algorithms that group data in 'hard' clusters, where each instance belongs to one and only one cluster, LDA allows each document to be a mix of topics, each with a certain probability. In the context of analyzing Airbnb reviews, each review is treated as a 'document' and the aim is to uncover the latent 'topics' that are present in these reviews. Topics in LDA are characterized by a distribution over a fixed vocabulary. For instance, a topic might consist of certain words such as 'clean', 'tidy', 'spotless', all with associated probabilities, and we might label this topic as 'Cleanliness'. if there are many reviews mentioning words like 'view', 'scenic', 'balcony', we might find these words constitute a prominent topic, which we could label as 'View'. For our objective of unsupervised learning where we want owners to know themes / topics which consumers care about, the output of LDA can be a useful tool for understanding the primary themes or topics that customers are discussing in their reviews. LDA can be an effective way to understand and categorize customer feedback on a large scale and therefore we chose it as our unsupervised learning technique.

Hyperparameter tuning: For Latent Dirichlet Allocation (LDA), as part of the hyperparameter tuning process, we expanded our list of stopwords based on your specific text data. By refining the list of stopwords, we tuned the input features the LDA model will use, which proved to have a significant impact on the model's performance and the interpretability of its output. Apart from the a list of common English words from nltk. Corpus. Stopwords. words("english"), we added additional stopwords such as 'los', 'angeles', 'flat', 'room', 'place', 'apartment', 'really', 'airbnb', 'good', 'nice', 'great', 'lovely', 'amazing', 'bnb', 'santa', 'monica', 'well', 'not', 'highly', 'thank', 'perfect', 'house', 'home', 'everything', 'ever', 'like', 'felt', 'loved', 'super', 'definitely', 'recommend', 'back', 'come', 'venice', 'even', 'day'. We removed these words from our textual reviews data because these words were overly common.

We transformed the textual data into a matrix representation via the Tfidf Vectorizer (Figure 13). Parameters were tactically adjusted to optimize the model's performance. 'Strip_accents' parameter was set to 'unicode' for character normalization, and 'ngram_range' was configured as (1, 2) to enable unigram and bigram extraction. The 'min_df' parameter was designated as 10, thus excluding terms with a frequency below this threshold. By setting 'use_idf' as 'True', inverse-document-frequency reweighting was enabled, which helps modulate term weights based on their frequency. The 'smooth_idf' was utilized to avoid zero divisions when a term doesn't exist in the corpus, and 'max_features' was capped at 5000 to limit the vocabulary size. These adjustments were made to enhance the accuracy of our topic modeling process.

```
count_vectorizer = TfidfVectorizer(
                        strip_accents='unicode',
                        preprocessor=None,
                        analyzer='word',
                        ngram_range=(1, 2),
                        min_df=10,
                        use_idf=True, smooth_idf=True,
                        max_features = 5000)
```

Figure13: Transforming textual data into a matrix representation.

Upon successfully transforming the text into a matrix representation, the matrix was fed into the LDA model employing the fit method. Parameters - including 'n_components' and 'learning_method' - were utilized to refine the model. The precise selection of the number of topics presented a significant challenge, as there is no ground truth and lacks a straightforward, efficient strategy to determine an

Num_Topics	Coherence_Score
2.0	0.542
4.0	0.597
6.0	0.639
8.0	0.611
10.0	0.598

Figure14: LDA coherence score sensitivity analysis

optimal number. To initiate the process, the 'n_components' was set to 6 as 6 topics gave the highest coherence score (Figure 14), subsequently we tried 10 since 10 gave the lowest perplexity score. Consequently, this step yielded a set of topics along with their respective distribution. We noted output from 6 topics rendered higher interpretability of the derived topics. By invoking '.components_' and 'get_feature_names()' methods, topics were sorted based on their distribution and results were subsequently displayed. Hyperparameter tuning steps catered to the effective execution of the LDA model.

Findings: The topics and term-frequencies printed were hard to interpret. So, we used pyLDAvis visualization (Figure 15) for better interpretation. Topic modeling using gensim on negative sentiment sub-dataset showed keywords such as 'dirty'. 'Towel', 'clean', 'bed', 'bathroom', 'toilet', 'cleaning', 'floor', 'sheet', 'hair', 'shower' which seem to highlight cleanliness issues! Topic can be labeled as 'Cleanliness Is Key' or 'Banish Bathroom / Bedroom Blunders'. Positive sentiment sub-dataset showed keywords such as 'clean', 'host', 'location', 'recommend', 'comfortable', 'easy', 'responsive', 'area', 'space', 'quiet', 'close' which we interpreted as key traits which makes one accommodation stand out from another. We can label this topic as "Responsive Host with Clean Quiet Convenient Space".

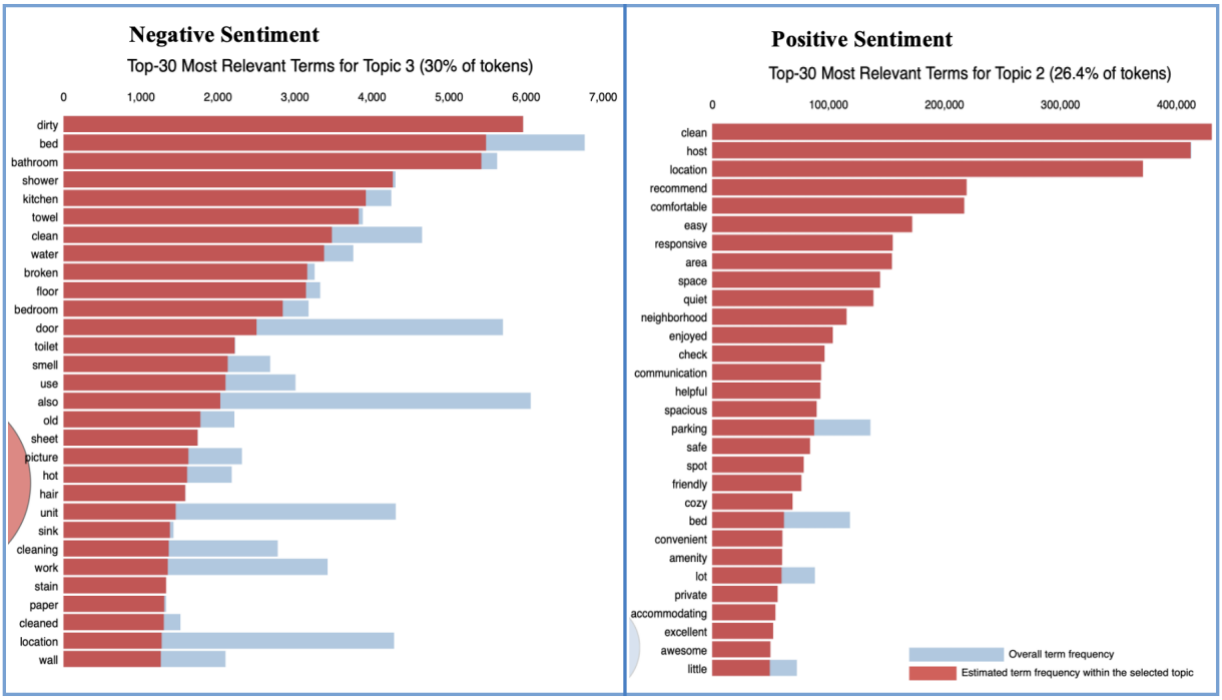


Figure15: LDA topic modelling using pyLDAvis visualization, refer to Appendix 3C for top 30 topics plotted against inter-topic distance map.

NMF

Non-negative Matrix Factorization (NMF) is a non-probabilistic linear algebraic model that factors high-dimensional vectors into a low-dimensionality representation. NMF takes advantage of the fact that the vectors often live in lower-dimensional spaces. NMF does not remove the mean of the column vectors (it does not center the data), and operates on the absolute values directly - hence the term 'non-negative'. This characteristic makes NMF particularly suitable and widely used for image data and text data, where negative components can be hard to interpret.

When applied to text data such as Airbnb reviews, NMF can help identify the latent topics/themes present in the reviews. In the context of NMF, reviews are represented as a matrix with one row per document and one column per token (e.g., word) appearing in the data set. NMF factors this matrix into two smaller matrices: a document-to-topic matrix, and a word-to-topic matrix that when multiplied together reconstruct the original matrix with minimal error. Each document and word are thereby associated with a vector of topic weights. For example, if the words 'clean' and 'tidy' often appear in the same reviews, they will likely be assigned to the same topic. Similarly, reviews that repeatedly use words like 'clean', 'tidy', 'neat' will also be assigned to this topic, indicating a high occurrence of cleanliness-related feedback. Thus, this topic could be interpreted as 'cleanliness'. On the other hand,

if words like 'view', 'balcony', 'scenery' frequently appear together, they might form a different topic, which we could interpret as 'scenic view'.

By analyzing the most relevant topics for each review, it's possible to get a sense of the main themes being discussed in the feedback by end consumers on any accommodation without having to read every review. In this way, NMF can be a powerful tool for summarizing and understanding large volumes of text data, and thus one of the chosen unsupervised learning methods the team has selected.

The NMF function receives three parameters: the corpus of textual data, the number of topics to be extracted, and a mapping from id to words from our dictionary. The Nmf object is initialized with a series of parameters designed to optimize the modeling process. The 'chunksize' parameter is set to 2000, indicating the number of documents to be used in each training chunk. This allows the algorithm to operate efficiently on larger corpora by breaking them down into manageable parts. The 'passes' parameter, defined as 5, indicates the number of passes through the corpus during the training. A higher number of passes can lead to a more accurate model at the cost of increased computation time. We chose 5 as it gave us good results in an acceptable amount of time. The 'kappa' parameter, set at 0.1, acts as a learning rate and makes the algorithm more robust against the choice of the initial topics. The 'minimum_probability' parameter is set at 0.01, filtering out topics that have a low chance of occurring, thus reducing noise in the output. The 'w_max_iter' and 'h_max_iter' parameters, set at 300 and 100 respectively, specify the maximum number of iterations for updating the 'w' and 'h' matrix in the algorithm. The 'w_stop_condition' and 'h_stop_condition' parameters, with values of 0.0001 and 0.001, act as termination conditions for the iteration process when the change in matrix elements falls below these thresholds. The 'eval_every' parameter is set to 10, meaning the model's performance is evaluated every 10 iterations to ensure the algorithm is improving. Each evaluation requires the model to pause training, assess its current performance, and then resume training, resulting in it being computationally expensive. Normalization is activated by setting 'normalize' to 'True'. This makes the algorithm more stable by ensuring that all features have the same scale. The above settings were done to ensure improvement to the model's accuracy and computational efficiency, as we apply the NMF method for our topic modeling process on Airbnb reviews.

Hyperparameter tuning: Stopwords for LDA were used for NMF for the purposes of hyperparameter tuning. Please refer back to the LDA description.

Findings: Based on the NMF topics, we found that not only does the experience living in the Airbnb matter but also does the proximity and convenience of the location itself. Additionally, the service from the host is important and so is the cleanliness of the unit. Some specific features like kitchens, bathrooms, bedrooms and parking are shown in the topics. These keywords also appear in the positive and negative comments. We interpret that these features are important for consumers and owners should focus on them to get good reviews. Achieving excellent scores in these domains may translate to enhanced guest experiences, better reviews, and ultimately a higher occupancy rate for hosts.

Figure16: NMF topic modelling using word cloud.

Evaluation of models

Choice of evaluation metrics: Coherence score and perplexity score were chosen as the two evaluation metrics. Reason as follows:

Coherence score- Coherence score measures the semantic similarity of high scoring words within each topic. A higher coherence score will mean that topics are more interpretable and meaningful. This is crucial in the context of Airbnb reviews since we want to extract tangible, understandable topics that can provide actionable insights for Airbnb hosts. For instance, if a topic relates to 'cleanliness', we would expect words such as 'clean', 'tidy', 'spotless' to appear together which highlights to the host that cleanliness is a key determinant between getting a good or bad review. A high coherence score indicates that our model effectively groups such related terms under one topic.

Perplexity score- Perplexity is a statistical measure of how well a probability model predicts a sample. In the context of LDA and NMF, a lower perplexity score implies the model is better at predicting new reviews on properties by consumers. However, as perplexity often has an inverse relationship with human interpretability such that models with lower perplexity don't necessarily yield more understandable topics. Despite this, perplexity still holds value in Airbnb review analysis as it provides an initial evaluation of the model's generalizing capability. A model with high perplexity might not be suitable as it implies the model is struggling to capture the underlying patterns in the review text.

In conclusion, while coherence focuses more on semantic interpretability while perplexity focuses on model prediction capability, the combination of both techniques offer a comprehensive evaluation mechanism. They help ensure our topic model not only generalizes well on unseen Airbnb reviews but also provides interpretable topics that can guide hosts in improving their services.

Sensitivity analysis: We have compared the results between the model with and without applying extensional stopwords. We found that the topics become more understandable after we applied extensional stopwords and have higher coherence scores. We also compared the result with different numbers of topics and found that increasing the number of topics makes it harder to summarize our topics. Also, we found that small adjustments in parameters could drastically change the results, showing the delicacy and complexity of topic modeling. NMF does not rely on probabilistic modeling, therefore perplexity, which is based on the likelihood of observing the test data, is not applicable to NMF.

Finding a tradeoff between coherence and perplexity that balances our approaches and fits our needs is key. We decided the number of topic for NMF based on the best coherence score. We decided to use coherence as the primary criterion for deciding on the number of topics for our model because higher coherence scores indicate more meaningful and coherent topics. As the number of topics increases, the model can capture more granular details, resulting in lower perplexity scores. We found a balance between interpretability (coherence) and performance (perplexity). Too few topics may result in less representative topics, while too many topics may lead to less interpretable results.

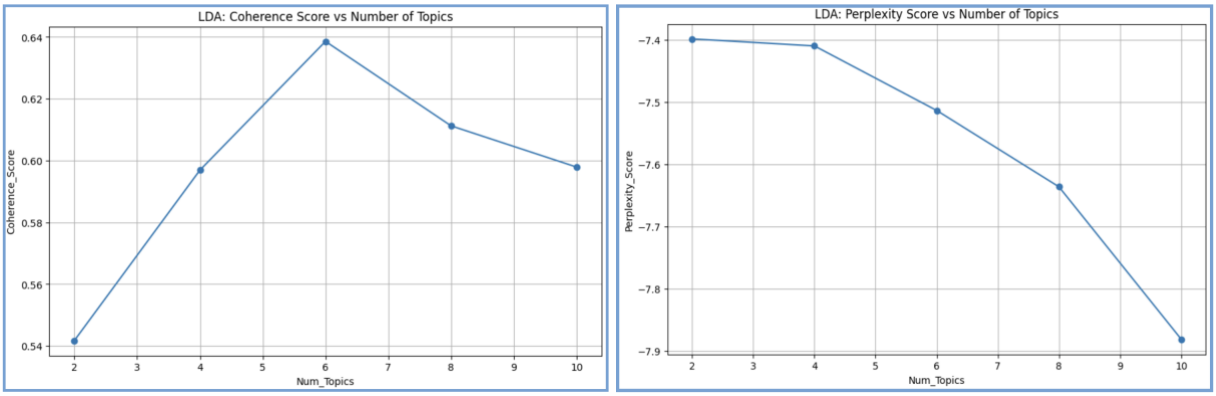


Figure17: Trade off analysis between coherence and perplexity score

Considering coherence and perplexity scores of our best model - LDA Figure 17, it appears that a model configuration with 10 topics provides the lowest perplexity score value while 6 topics provide the highest coherence score. Prioritizing perplexity optimization may not guarantee topics that are easily interpretable to humans. rendering the emphasis on perplexity less critica when compared to coherence score. The interpretability

of the derived topics—a crucial component for human comprehension and practical utility—is not adequately served at this elevated topic count. Our Coherence Score analysis supports a selection of 6 topics as an optimal balance between quantitative metrics and meaningful interpretability. Further experimentation, with more than 10 topics, led to a degradation in the coherence and intelligibility of the generated topics, reaffirming our decision to favor a model with 6 topics.

SUMMARY

Some topics were similar for LDA and NMF, for example, clean, host, and location can be found in both models. We also found the main features from listings.csv by using absolute loadings with PCA and NMF (charts and results are available in appendix). By comparing the results between numerical and textual data analysis, keywords appear on both sides. This shows that topic modeling and finding main features are heavily nuanced and requires human interpretation.

Comparing LDA and NMF revealed notable differences in the computational demands (Figure 18). LDA exhibited superior performance with regard to topic coherence (0.639), securing a marginally higher score than NMF (0.593). Moreover, LDA proved to be more time-efficient, completing the local model run in approximately 2.5 hours, in contrast to the 26.5 hours required for NMF. On the strength of these metrics—improved time-efficiency and superior coherence score—we opted for the LDA model. Referring to Figure 17, an inspection of the LDA's Perplexity Score chart reveals that lower values correspond to a model that is better equipped to handle new, unseen data, thus further underscoring our model choice.

Comparing LDA against NMF		
	Coherence score	Time taken
LDA	0.639	2 hrs 30 min
NMF	0.593	26 hrs 30 min

Figure18: Comparing LDA against NMF

DISCUSSION

From performing Part B, we learned that machine learning is an iterative process. We learned that constructing effective machine learning models doesn't occur instantaneously, but unfolds progressively through a series of repeated refinements and code optimizations.

What surprised you about your results?: Our analysis using LDA and NMF delivered several unexpected outcomes. Firstly, the degree of topic overlap in LDA was surprising. We anticipated some level of redundancy given the nature of natural language, but the amount of repetition we observed surprised us. NMF model on the other hand surprised us with the complexity of its topic groupings. Some of the top words in a topic seemed loosely connected, requiring a deeper level of interpretation from domain experts to discern their common thread. Both models' sensitivity to hyperparameters was an eye-opener. Small adjustments in parameters, such as the number of topics, could drastically change the results, indicating the complexity and delicacy of topic modeling.

Challenges and responses: One of the main challenges was the difficulty in comparing results from LDA and NMF due to varying numbers of topics. In the LDA model, there was repetition of topics, making it hard to differentiate them, whereas, in the NMF model, grouping top words into coherent topics was challenging. Within the LDA model, we observed a redundancy in topics, which obscured the distinctness of each theme.

Simultaneously, the NMF model presented its own unique challenge: it was not straightforward to categorize the most prominent words into coherent topics. Adding to the complexity was the fact that, unlike supervised learning, there was no available 'ground truth' against which we could contrast our findings. This called for inventive problem-solving, prompting us to resort to hyperparameter tuning. This involved additional stop words to refine the results. As a part of our solution, we employed our own judgment, meticulously reviewing each topic model to ascertain if it was coherent and meaningful. This iterative process enabled us to navigate the challenges posed by the inherent uncertainty in unsupervised learning.

Further work: Given more time and computing resources, we would like to use BERT to create meaningful vector representations of our text data, which can then be used with traditional clustering algorithms to identify the underlying topics. BERT has the potential to capture more complex patterns and could provide better results, but would require significantly more computational resources and potentially more data. Embeddings generated by BERT have a high dimensionality (768 dimensions for the base model). To make the subsequent steps tractable, we can use techniques like PCA (Principal Component Analysis) to reduce the dimensionality of these embeddings. It would be interesting to see how we can use PCA with BERT to improve results of topic modeling.

ETHICAL CONSIDERATIONS

Given the nature of our project with Airbnb, we made every effort to ensure ethical handling of data and mitigating potential adverse impacts. Our primary focus was to maintain privacy and confidentiality. Although our dataset comprised publicly accessible reviews, we were mindful of potentially personally identifiable information (PII) and made sure to not include any such data in our machine learning mode. Despite the context of our project not dealing with any protected classes (race, religion, sex, age, etc.), our vigilance towards ethical considerations was not compromised. We completed a third-party impact test: the Canadian Government's Algorithmic Impact Assessment and scored a relatively low impact score indicating the ethically sound nature of our project.

In terms of potential class impacts, we were cognizant of the impacts our findings may have on Airbnb hosts who might not have as many reviews or those hosts with more negative reviews in the event our supervised learning model was deployed. Such hosts might feel disadvantaged as they may appear less favorable to potential customers based on our sentiment analysis. Hence, we advise all stakeholders to interpret the findings in the broader context of guest experiences, which are inherently subjective and can vary widely.

Machine learning models in the hospitality and tourism industry, while beneficial, may inadvertently introduce the issue of Secondary Use (Veerle.V, 2011). This refers to using data obtained for one purpose (such as customer reviews for feedback) and using it for an unrelated purpose (such as evaluating property quality) without explicit consent.

Though the reviews are publicly available, the aggregated insights might influence decision-making processes for both guests and hosts. This secondary use of data could potentially lead to unintended biases in the Airbnb ecosystem. Furthermore, as machine learning continues to be integrated into various industries, there is an increasing need for transparency and recourse mechanisms. If inaccuracies arise from our model predictions, it is vital to provide a method for redress, especially considering the potential influence of these ratings on hosts' livelihoods. We advocate for a system where hosts can challenge or provide additional context to reviews that they feel might not present a complete picture of their property. As data scientists, we echo the call for the industry to rally together in shaping a data policy that safeguards personal information and ensures fairness. Just as the AIS was developed for public safety, the sharing economy industry can also strive to establish a similar framework for public good. In the era of big data, our responsibility for ethical considerations in our research and its potential implications remains a top priority.

STATEMENT OF WORK

Chi Huen Fong: Language Detection, Data Cleaning & Preprocessing, Unsupervised Learning

Steven Ong: Sentiment Tagging, Supervised Learning - Traditional Models, NLP - BERT & Distilled BERT, Unsupervised Learning – LDA, Maintaining Github Repository

John Kaspers: Unsupervised Learning - NMF, Unsupervised Learning - coherence and perplexity

All members: Content Writing and Design for Project Proposal, Standup Videos and Final Report

References

Meijerink J, Schoenmakers E. (2019). Why are online reviews in the sharing economy skewed toward positive ratings? Linking customer perceptions of service quality to leaving a review of an Airbnb stay. Retrieved May 27, 2023 from https://www.researchgate.net/ publication/ 340538870 _Why_are_online _reviews_ in_the_ sharing _ economy_skewed_toward_positive_ratings_Linking_customer_perceptions_of_service_quality_to_leaving_a_r eview_of_an_Airbnb_stay

Georgios. Z, Proserpio. D, Byers J. (2020). A First Look at Online Reputation on Airbnb, Where Every Stay is Above Average. Retrieved May 27, 2023 from https://people.bu.edu/zg/publications/airbnbreviews.pdf

Pohle, A. (2023). Four Stars for Peeling Paint and Broken Doors? What's Behind High Airbnb Ratings. Retrieved May 27, 2023 from https://www.wsj.com/articles/four-stars-for-peeling-paint-and-broken-doors-whats- behind - high-airbnb- ratings-da26390e

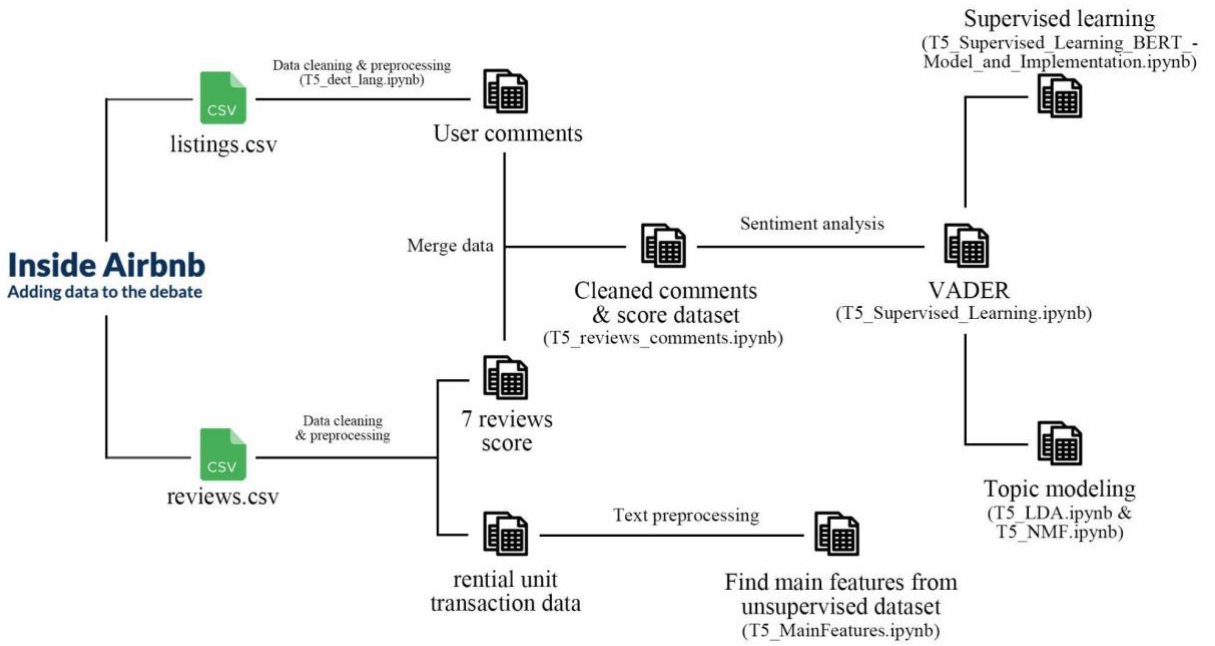
Karl H., Victor C., Chrisina J. (2022). A review on Natural Language Processing Models for COVID-19 research. Retrieved Jun, 15, 2023 from https://www.sciencedirect.com/science/article/pii/S2772442522000326

Veerle V, Louise C, Matthew W, Libby B, Laurence H. (2011). Managing and Sharing Data, Best Practice for Practitioners. Retrieved Jun, 16, 2023 from https://dam.ukdataservice.ac.uk/media/622417/managingsharing.pdf

APPENDIX

Github Repository - https://github.com/stevienovak/M2-Intriguing-AirBnB-Text-ML_NLP.git

Appendix 1A



Appendix 2A

Logistic Regression

Testing Set
Accuracy: 0.5591642636717762

	precision	recall	f1-score	support
0	0.5805	0.6098	0.5948	7943
1	0.4741	0.4887	0.4813	7954
2	0.4557	0.4122	0.4328	7878
3	0.4965	0.4542	0.4744	7908
4	0.7572	0.8325	0.7931	7851
accuracy			0.5592	39534
macro avg	0.5528	0.5595	0.5553	39534
weighted avg	0.5525	0.5592	0.5550	39534

Naive Bayes

Testing Set
Accuracy: 0.41389689887185716

	precision	recall	f1-score	support
0	0.5778	0.3349	0.4240	7943
1	0.4189	0.4345	0.4266	7954
2	0.3379	0.3244	0.3310	7878
3	0.3153	0.2939	0.3042	7908
4	0.4570	0.6836	0.5478	7851
accuracy			0.4139	39534
macro avg	0.4214	0.4143	0.4067	39534
weighted avg	0.4215	0.4139	0.4066	39534

Random Forest

Testing Set
Accuracy: 0.559847220114332

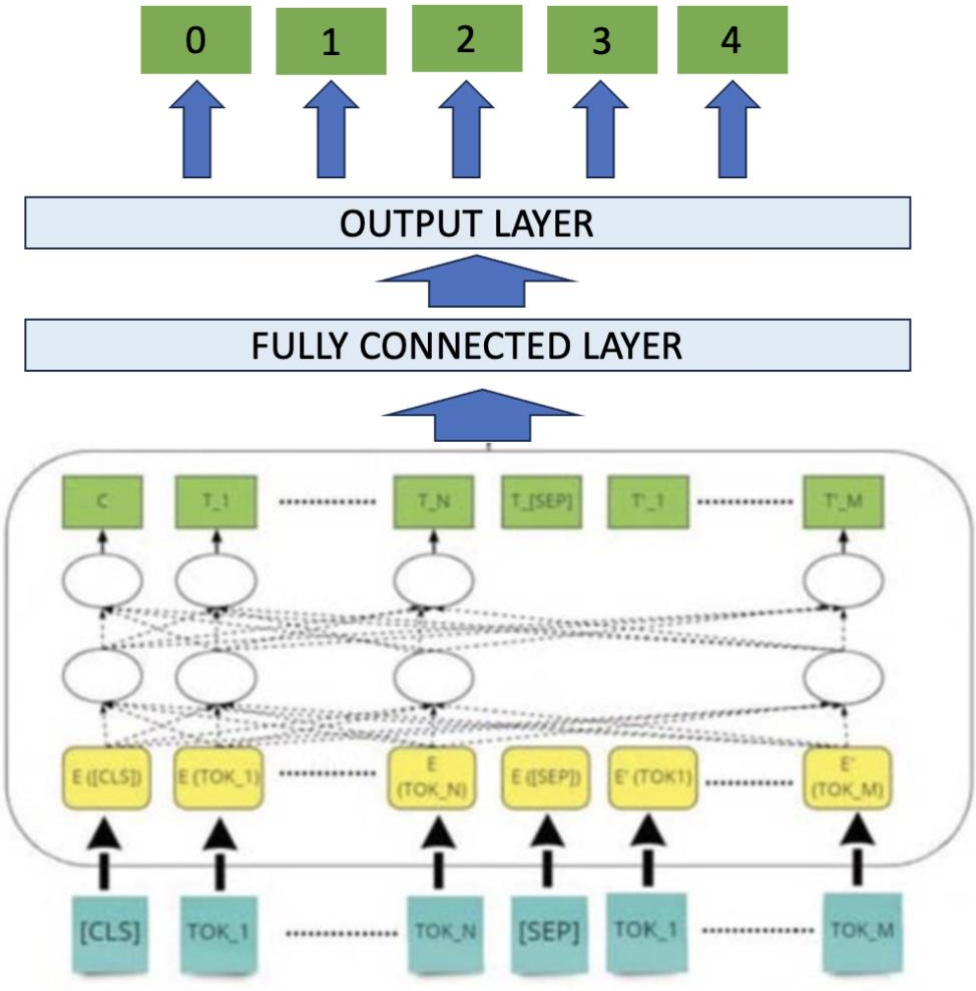
	precision	recall	f1-score	support
0	0.6526	0.6190	0.6354	7943
1	0.5646	0.4669	0.5111	7954
2	0.5220	0.4510	0.4839	7878
3	0.4870	0.3423	0.4020	7908
4	0.5546	0.9224	0.6927	7851
accuracy			0.5598	39534
macro avg	0.5562	0.5603	0.5450	39534
weighted avg	0.5563	0.5598	0.5449	39534

Linear Support Vector Machine

Testing Set
Accuracy: 0.3178529873020691

	precision	recall	f1-score	support
0	0.7188	0.2391	0.3588	7943
1	0.5098	0.2345	0.3212	7954
2	0.3534	0.0722	0.1199	7878
3	0.2232	0.0582	0.0923	7908
4	0.2629	0.9901	0.4155	7851
accuracy			0.3179	39534
macro avg	0.4136	0.3188	0.2616	39534
weighted avg	0.4143	0.3179	0.2616	39534

Appendix 2B



Input and output pattern of BERT model for downstreaming tasks. A fully connected layer is used to find the output. C1, C2, C3 are 3 classes which represent positive, negative, and neutral reviews respectively. TABLE I. MULTI-LABEL CLASS FOR THE DATASET Encoded Value (Y) Output Class 0- Bad, 1 -Nay, 2 - Average, 3 - Good, 4 - Awesome!!

Appendix 3A

Positive WordCloud (old stopwords)

Neutral WordCloud(old stopwords)

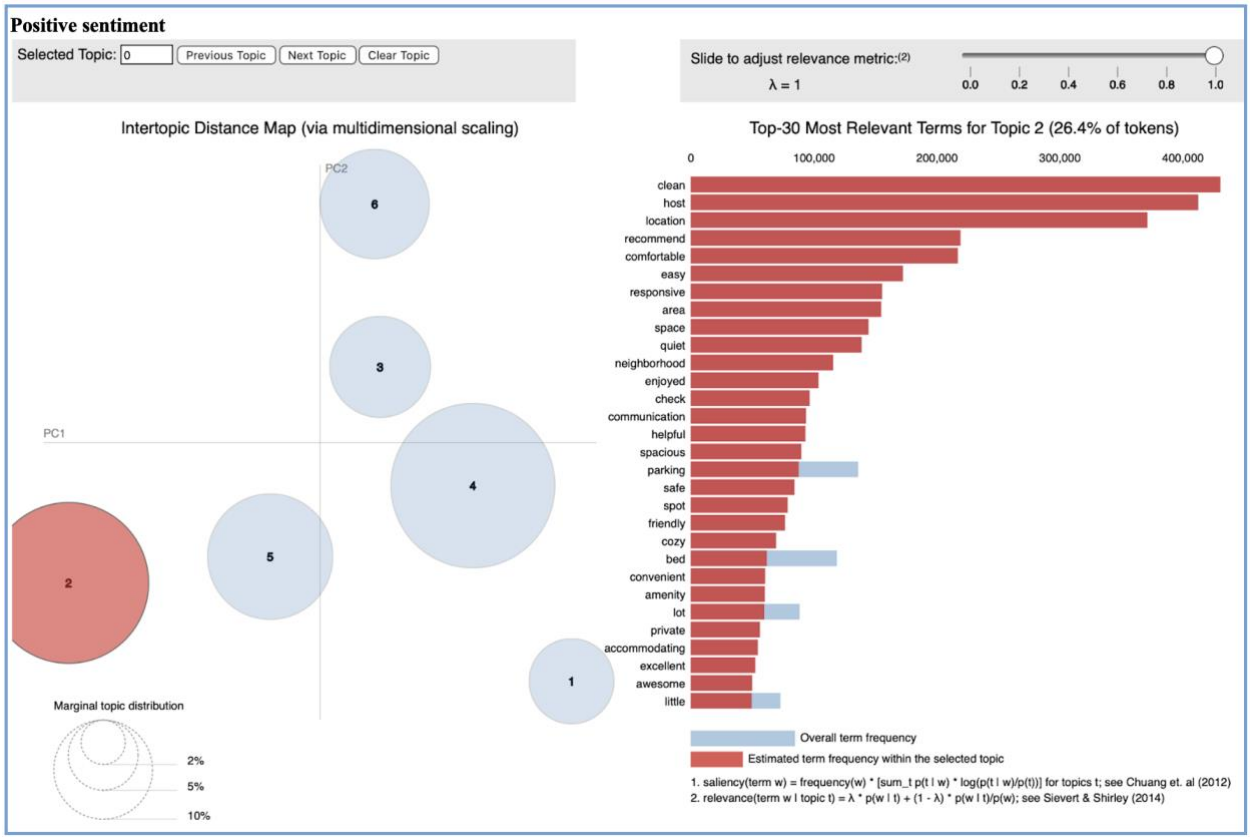
Negative WordCloud(old stopwords)

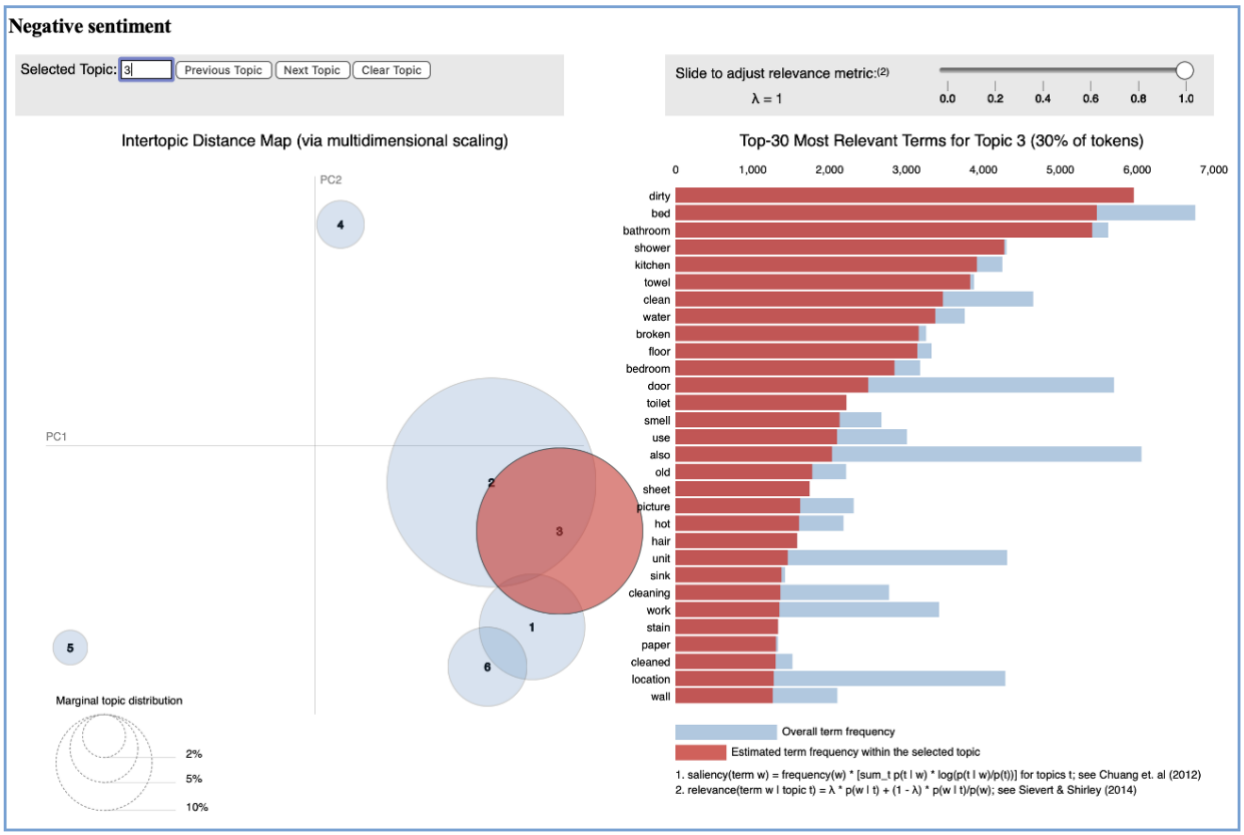
Appendix 3B

LDA TABLE with old stopwords

Positive	Negative
Topic #1: location clean host recommend communication responsive comfortable excellent highly recommend highly Topic #2: house convenient convenient location forward view guest location expected guest house look Topic #3: not clean house location everything parking easy can comfortable space Topic #4: home recommend definitely beautiful staying host time felt clean like Topic #5: well always equipped stocked light tidy well equipped well stocked kitchen communicating Topic #6: distance walking walking distance quiet neighborhood parking clean easy safe quiet neighborhood	Topic #1: parking horrible street difficult street parking park block away find noisy Topic #2: worst ever experience worst experience air conditioner conditioner experience ever air canceled host Topic #3: cozy bad ok everything location ant policy cable cheap life Topic #4: not dirty bad clean location bathroom neighborhood terrible bed recommend Topic #5: wouldnt recommend wouldnt recommend perfect spot convenient parking spot location quick spacious Topic #6: accommodating ill disappointing look house rule son condo back guy

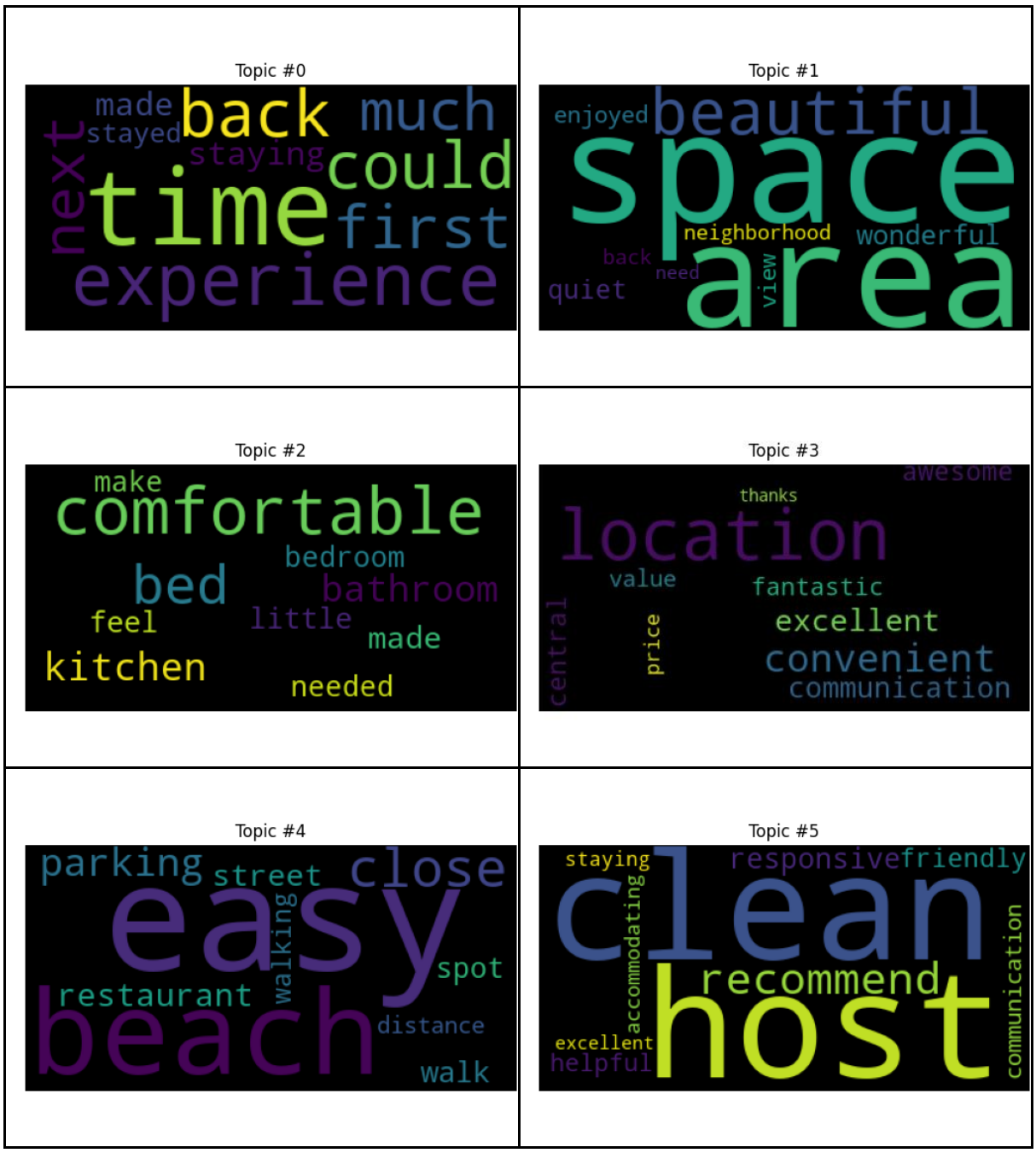
Appendix 3C


Positive sentiment
Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 2 (26.4% of tokens)
0 100,000 200,000 300,000 400,000
clean
host
location
recommend
comfortable
easy
responsive
area
space
quiet
neighborhood
enjoyed
check
communication
helpful
spacious
parking
safe
spot
friendly
cozy
bed
convenient
amenity
lot
private
accommodating
excellent
awesome
little
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)


Negative sentiment
Selected Topic: 3
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 3 (30% of tokens)
0 1,000 2,000 3,000 4,000 5,000 6,000 7,000
dirty
bed
bathroom
shower
kitchen
towel
clean
water
broken
floor
bedroom
door
toilet
smell
use
also
old
sheet
picture
hot
hair
unit
sink
cleaning
work
stain
paper
cleaned
location
wall
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Appendix 3D

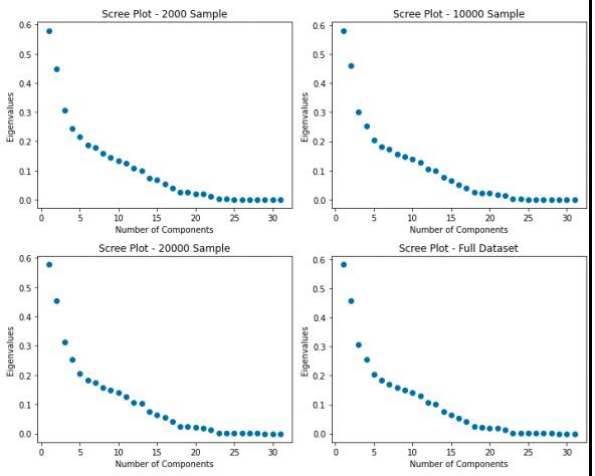
NMF Word Clouds by topic:



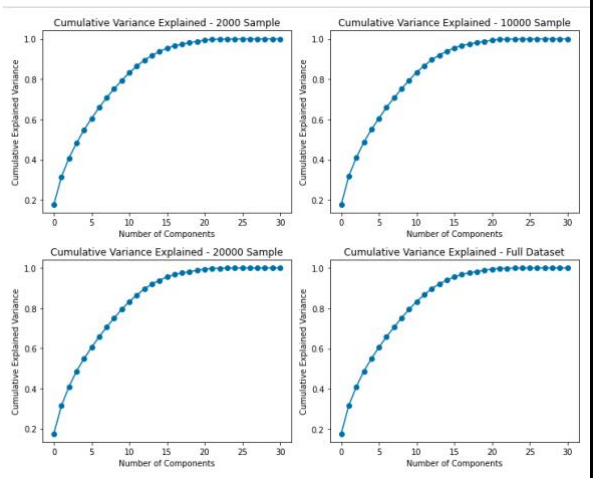
Appendix 3E

absolute loadings with PCA results to find main features in unsupervised learning:

Scree Plot for number of topic in 4 different dataset size. Best component is 5.



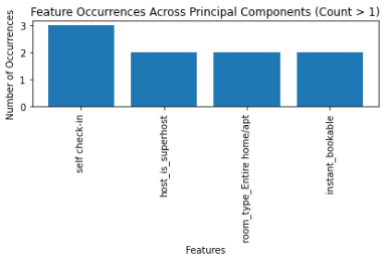
Cumulative variance explained in 4 different dataset size.



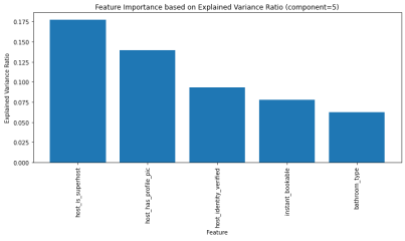
Result in absolute loadings of component = 5 and 16 (60% vs 95% of explained variance). Both of them are very similar.

	Component=5		Component=16	
	host_is_superhost	host_has_profile_pic	host_is_superhost	host_has_profile_pic
0	0.315586	0.009898	0.315586	0.009898
1	0.272396	0.001228	0.272397	0.001228
2	0.062564	0.010909	0.062565	0.010908
3	0.199575	0.010592	0.199576	0.010591
4	0.441024	0.003093	0.441053	0.003099

Count the features occurrences across different components due to the low variance covered by each component. Self check-in appeared the most, followed by host is super host, entire home room type and instant bookable.



Calculate the variance explained ratio to gain insight into which features contributed most to the overall variance. Host information covered the largest ratio of explained variance.

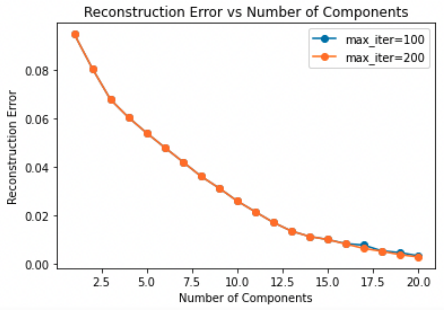


Absolute loadings summary:

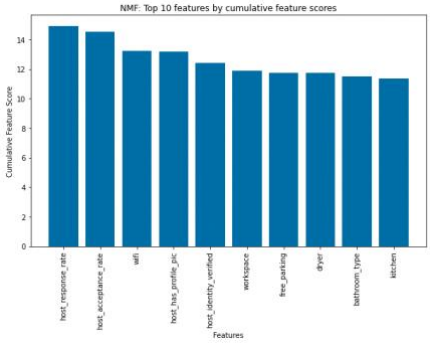
Combining the findings of these two results, we believe that host and convenience are the most important factors that users consider when booking.

Appendix 3E

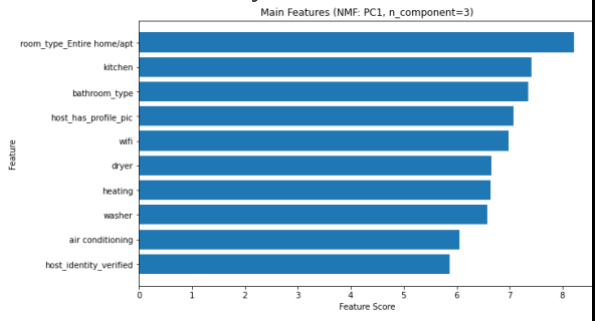
Evaluate the model performance with hyperparameter tuning (max_iter and number of component).



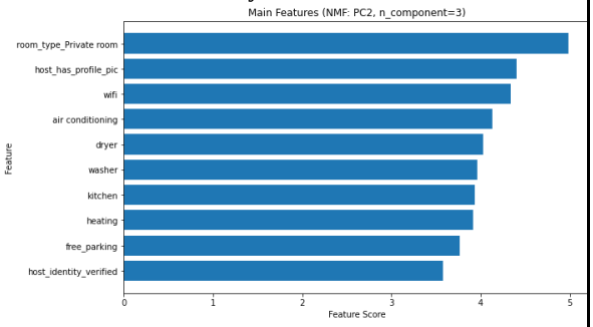
Features with highest feature score across the whole dataset. Host information and wifi have the highest cumulative feature score.



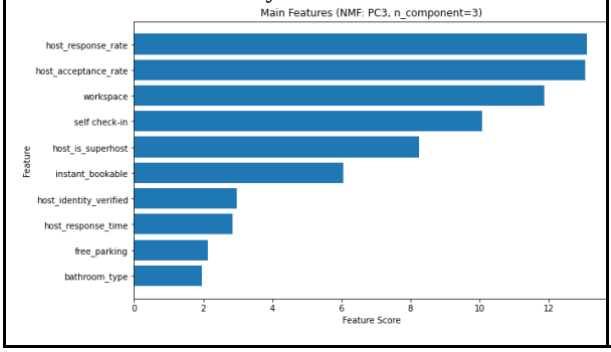
Main features in PC1 by feature score.



Main features in PC2 by feature score.



Main features in PC3 by feature score.



NMF summary:

Multiple features consistently appeared as the top ranked features across multiple components, such as host has profile picture, kitchen, wifi, dryer, air conditioning and washer.

Based on the result, we found that host related attributes, wifi, dryer, free parking, workspace and kitchen were the most important.