1. Abstract

Understanding customers' sentiments has great importance in today's business world. It can help companies to learn how customer feel about their products and then improve them accordingly. In this project, we use several machine learning methods (including Logistic Regression, Random Forest, Naïve Bayes, Gradient Boosting) to estimate customers' recommendation (whether recommend product or not) through review text. And utilize 3 methods (including adding new features, adjusting stop words list, and hyperparameter tuning) to improve the performance of models. Finally, two conclusions are drawn: in terms of models, Logistic Regression and Naïve Bayes perform best; and in business applications, our model can identify and track bad suggestions to improve the company's product and/or service.

content

2. Data Description

In this project, we use a real business world dataset which includes 23486 rows and 10 feature variables. In this dataset, each row corresponds to a customer record, including variables as following table.

Table 1. Variable Name and Interpretation

Variable Name	Interpretation
Clothing ID	Integer categorical variable that refers to the specific piece being reviewed
Age	Positive integer variable of the reviewers age
Title	String variable for the title of the review.
Review Text	String variable for the review body.
Rating	Positive ordinal integer variable for the product score granted by the customer from 1 Worst to 5 Best
Recommended IND	Binary variable stating where the customer recommends the product where 1 is recommended, 0 is not recommended
Positive Feedback Count	Positive integer documenting the number of other customers who found this review positive
Division Name	Categorical name of the product high level division
Department Name	Categorical name of the product department name
Class Name	Categorical name of the product class name

3. Basic Exploratory Data Analysis

3.1 Review Distribution Per Division and Per Department

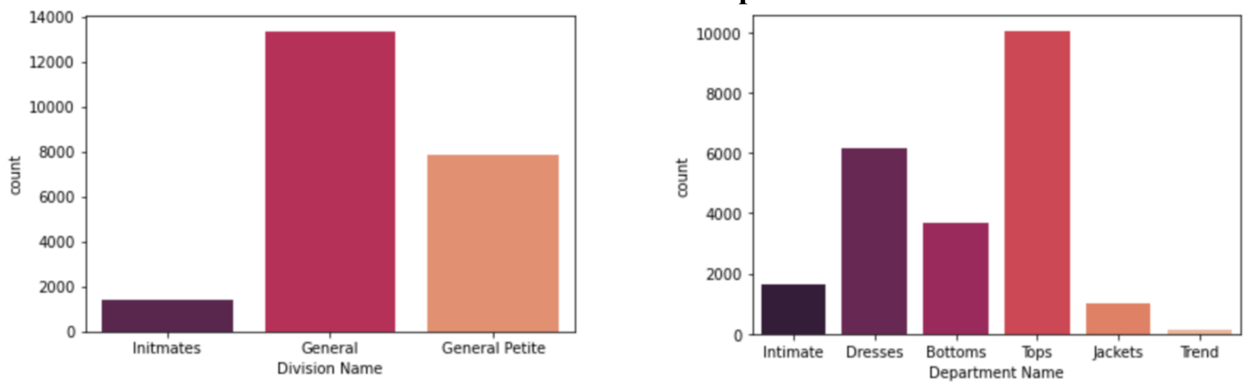


Fig.1 The frequency distribution of apparels per division and department

Figure1 shows the review distribution in different division and department. This can give the company an insight on which products are most popular with customers.

3.2 Department Name and Rating

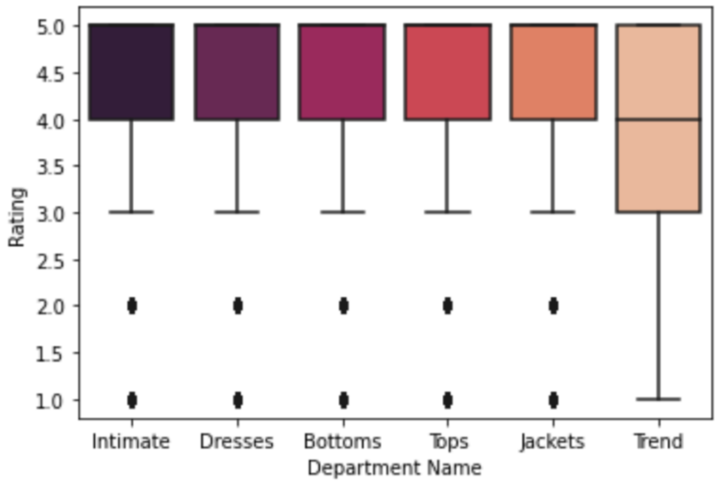


Fig.2 The frequency distribution of apparels per division and department

From this figure, we can see that the ratings (product score) of Intimate, Dresses, Bottoms, Tops, Jackets granted by customers are relatively concentrated, indicating that the evaluations are relatively consistent. However, customer ratings for Trend fluctuate greatly, indicating that different customers have different definitions of population.

3.3 Recommended IND and Rating

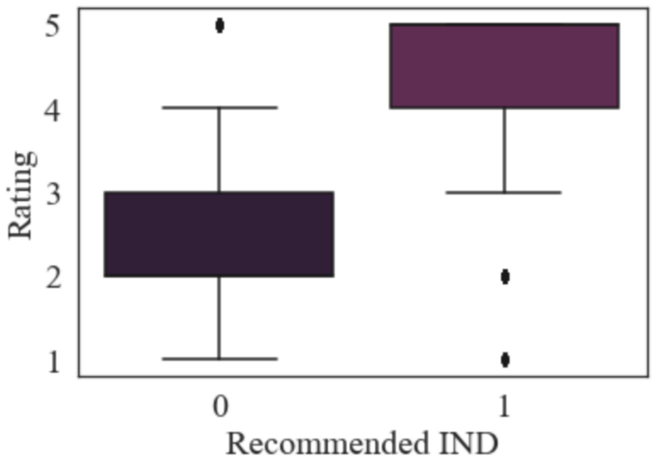


Fig.3 The relationship between Recommended IND and Rating

Customers who would like to recommend gave a higher rating in general and vice versa. The above figure also confirms the relationship. Therefore, in this report, we choose to analyze recommendations only.

4. Review Text & WorldCloud

4.1 Review Text

In this project, ‘Review Text’ is the most important variable for us to analyze. Unlike instant data, human language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text.

Therefore, before we start predicting, we try to make good use of Natural language processing (NLP) to translate text from one language to another to get that text in a format that the machine learning algorithms can understand. There are mainly 4 steps used to process our review text data. (Refer to figure 4) And in the following part, this report would explain each step in detail.

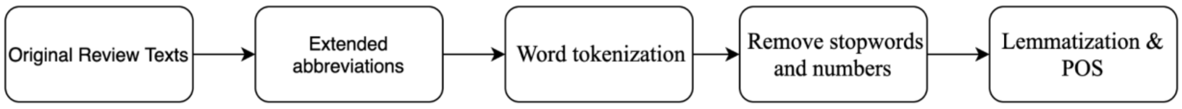


Fig4: Steps to process Review Texts

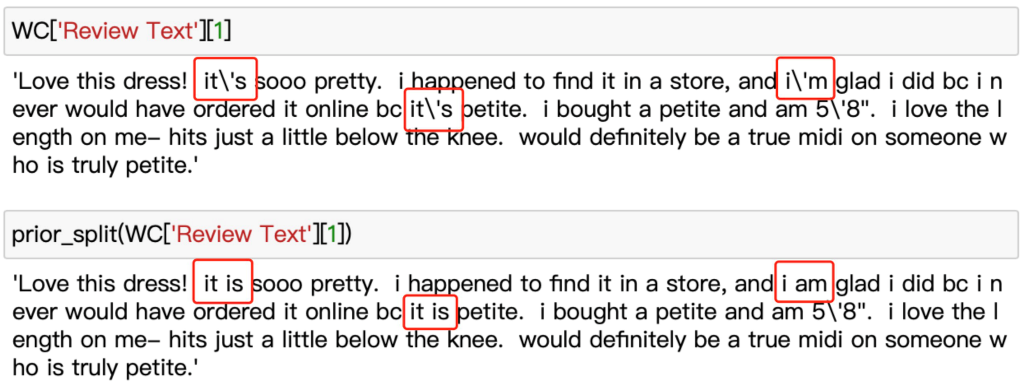
4.1.1 Extended abbreviations – Text standardization

After observing all review texts, we found that there are some common abbreviations like I’m, it’s, which may cause trouble for NLP. So we turn them into full words, like I am, It is.

- Code

```
def prior_split(data):
 data=re.sub("n't",'not',data)
 data=re.sub("i'm",'i am',data)
 data=re.sub("I'm","I am",data)
 data=re.sub("it's","it is",data)
 data=re.sub("It's","It is",data)
 return data
```

- Example



```
WC['Review Text'][1]
```

```
'Love this dress!  it\'s sooo pretty.  i happened to find it in a store, and i\'m glad i did bc i n
ever would have ordered it online bc it\'s petite.  i bought a petite and am 5\'8".  i love the l
ength on me– hits just a little below the knee.  would definitely be a true midi on someone w
ho is truly petite.'
```

```
prior_split(WC['Review Text'][1])
```

```
'Love this dress!  it is sooo pretty.  i happened to find it in a store, and i am glad i did bc i n
ever would have ordered it online bc it is petite.  i bought a petite and am 5\'8".  i love the l
ength on me– hits just a little below the knee.  would definitely be a true midi on someone w
ho is truly petite.'
```

Fig5: Extended abbreviations code and example

4.1.2. Word tokenization

The process of breaking up sentences into their constituent words.

Here, we define a function to decide our split method, which is "Match one or more non-letters for cutting. The matched non-letters are not cached".

```
def split_method(data):
 split_word=re.split('\W+',data)
 return split_word
```

Fig 6: Word tokenization code

4.1.3. Remove stopwords and numbers

(1) To save storage space and improve search efficiency, NLP automatically filters out certain words. Suc as some function words "the", "is" and so on. After downloading the stopwords from 'nltk', we also do some process to make it better to process word clouds that we need later. Firstly, we drop the last 36 stopping worlds like wouldn't. Secondly, we remove 'not' from the stopwords list because we found that it could represent customers' attitudes. Lastly, we extend "would", "could" to the list because those words appear frequently in our review text but have no special meaning.

(2) Also, numbers were also dropped for creating a better worldcloud later.

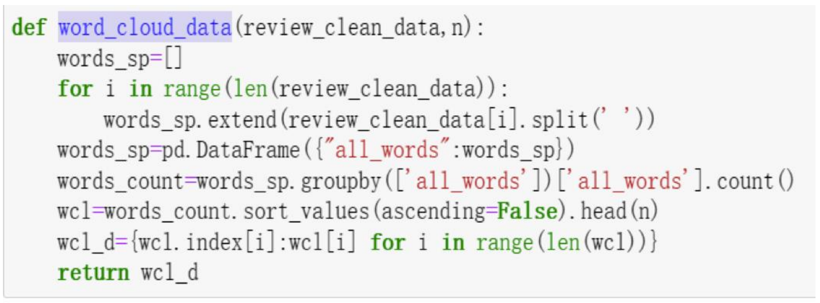
4. 1.4 Lemmatization

Lemmatization is an important part of text preprocessing, which is similar to stem extraction (stemming). We use the function in nltk to achieve it.

4.2 Word Cloud

After the completion of the previous processing, we save the results of our translation of "Text Review "in a new variable called "review clean", and use this variable to calculate the word frequency and draw the word cloud maps in different dimensions. By introducing wordcloud libraries, we get the popular words described by customers intuitively in several ways.

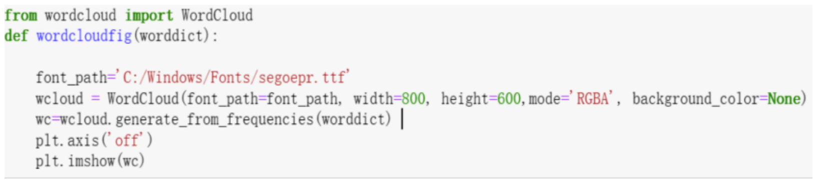
We firstly define a function called "word_cloud_data", whose main task is to further splitting the words in "review clean" and then calculate the word frequency:



```
def word_cloud_data(review_clean_data,n):
    words_sp=[]
    for i in range(len(review_clean_data)):
        words_sp.extend(review_clean_data[i].split(' '))
    words_sp=pd.DataFrame({"all_words":words_sp})
    words_count=words_sp.groupby(['all_words'])['all_words'].count()
    wcl=words_count.sort_values(ascending=False).head(n)
    wcl_d={wcl.index[i]:wcl[i] for i in range(len(wcl))}
    return wcl_d
```

Fig7: word_cloud_data function

Then, we continued to define a new function called 'wordcloudfig', which enables us to apply wordcloud library to draw wordcloud maps:



```
from wordcloud import WordCloud
def wordcloudfig(worddict):

    font_path='C:/Windows/Fonts/segoepr.ttf'
    wcloud = WordCloud(font_path=font_path, width=800, height=600,mode='RGBA', background_color=None)
    wc=wcloud.generate_from_frequencies(worddict)
    plt.axis('off')
    plt.imshow(wc)
```

Fig8: wordcloudfig function

4.2.1 The first 100 words that are most commonly used in the whole review

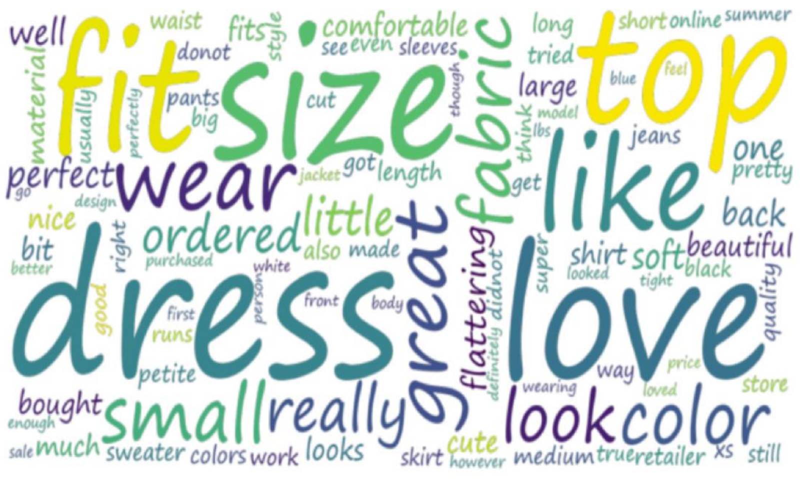
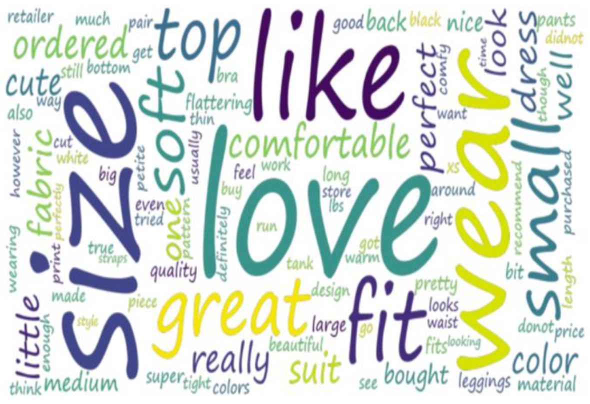


Fig9: Wordcloud for the whole review

From this figure, we can easily see that the words used most frequently by users are: dress, love, size, top, fit and so on. It can be seen that color, size and fabric are the most common things that have been talked about by customers. People are more likely to use verbs and adjectives such as love, like, great and perfect to directly express their satisfaction with the goods they buy.

4.2.2 The first 100 words that are most commonly used for different produc



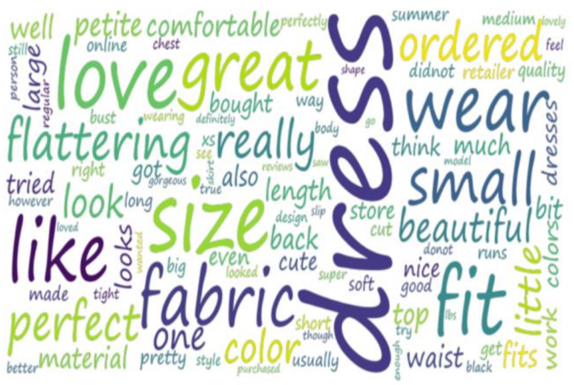


Fig10: 'Intimate' Department's words Fig11: 'Dresses' Department's words

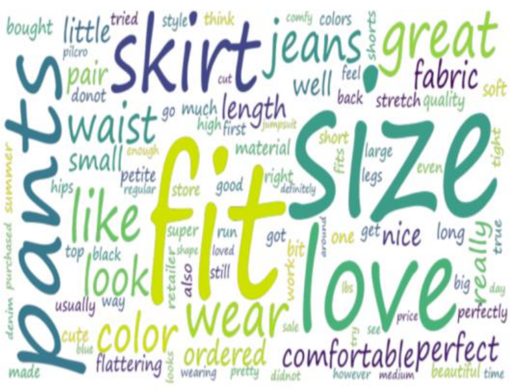


Fig12: ‘Bottoms’ Department’s words

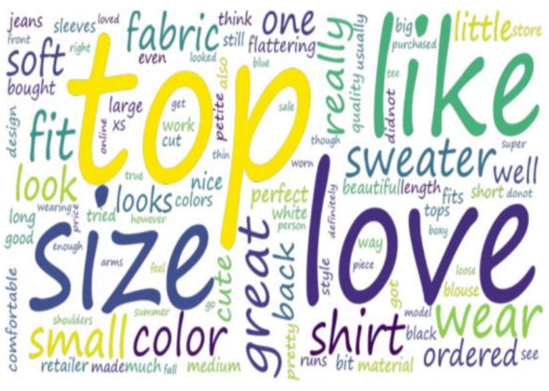


Fig 13: ‘Tops’ Department’s words

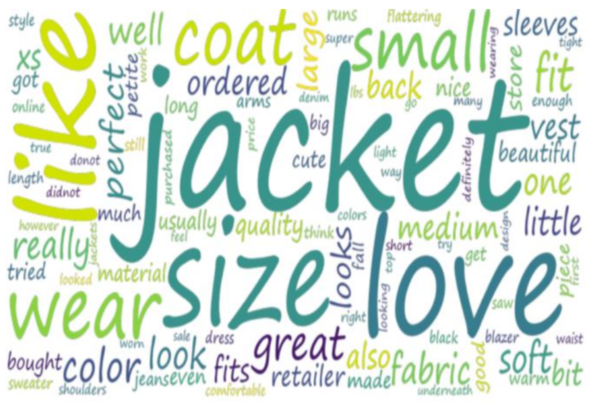


Fig14: ‘Jackets’ Department’s words

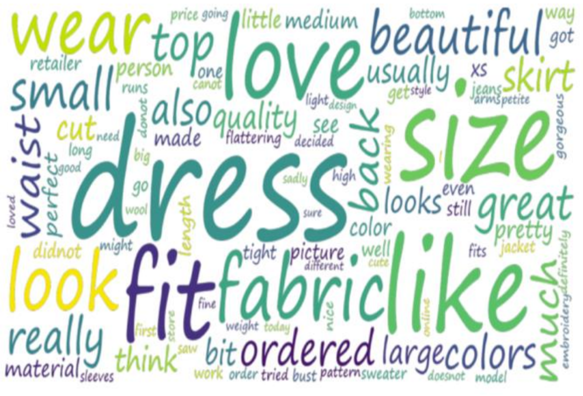
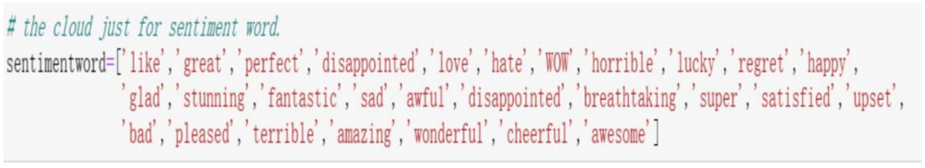


Fig15: ‘Trend’ Department’s words

There are 6 department categories: ‘Tops’, ‘Dresses’, ‘Bottoms’, ‘Intimate’, ‘Jackets’ and ‘Trend’. For each category, we draw a word cloud to see whether there are some differences about customers’ reviews. Customers’ reviews for different departments are not that similar. ‘Tops’, ‘Intimate’ and ‘Jackets’ are more common with the key word ‘soft’. While ‘beautiful’ is more special for the ‘Trend’ department.

4.2.3 The first 100 words that are most commonly used for customers sentiments

To be able to accurately extract sentiment words from user comments, we examined the comments very carefully, extracted the most frequently occurring sentiment words, and stored them in a list called ‘sentimentword’:



```
# the cloud just for sentiment word.
sentimentword=['like','great','perfect','disappointed','love','hate','WOW','horrible','lucky','regret','happy',
               'glad','stunning','fantastic','sad','awful','disappointed','breathtaking','super','satisfied','upset',
               'bad','pleased','terrible','amazing','wonderful','cheerful','awesome']
```

Fig 16: sentimentword

Then, in the same way, we mapped the word cloud of the most frequently used emotion words by users:

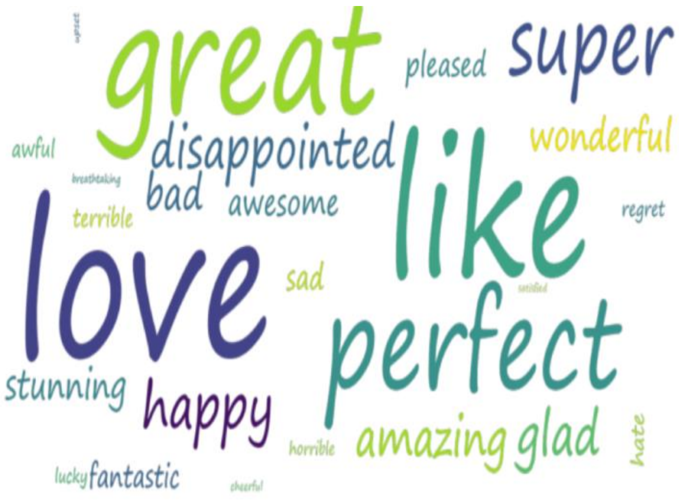


Fig 17: wordcloud for sentiment words

We can see that most of the top 10 words are positive sentiments except for a ‘disappointed’ one.

5. The Model Building

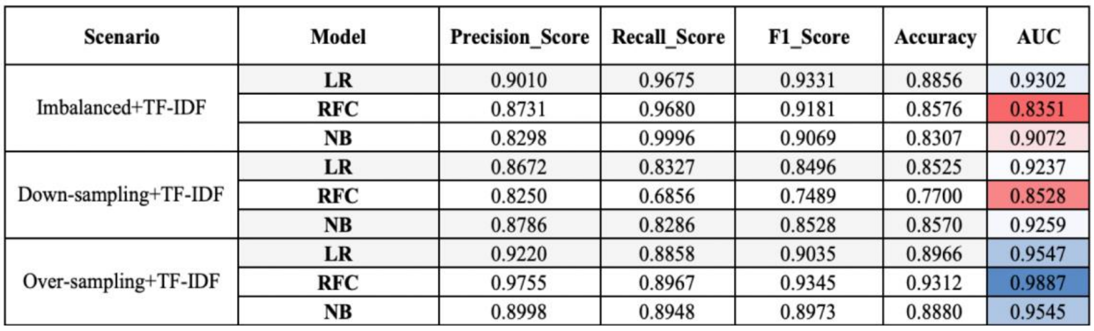
5.1 Model Preprocessing

For further data modeling, we first need to preprocess the model, in which the main tasks are: to transform the text data into vectors and resample the data for the problem of data imbalance.

5.1.1 Imbalance

When we analyzed the data, we found that our target variable “Recommended IND” had a very significant data imbalance. In our half-way report, we find that the AUC of models with undersampling data and oversampling is quite close. But with undersampling, the dataset can be much smaller, which enables us to save more time when building models. Therefore, we directly used an undersampling method to adjust the data here.

Table 2: Performance for different models



Scenario	Model	Precision_Score	Recall_Score	F1_Score	Accuracy	AUC
Imbalanced+TF-IDF	LR	0.9010	0.9675	0.9331	0.8856	0.9302
	RFC	0.8731	0.9680	0.9181	0.8576	0.8351
	NB	0.8298	0.9996	0.9069	0.8307	0.9072
Down-sampling+TF-IDF	LR	0.8672	0.8327	0.8496	0.8525	0.9237
	RFC	0.8250	0.6856	0.7489	0.7700	0.8528
	NB	0.8786	0.8286	0.8528	0.8570	0.9259
Over-sampling+TF-IDF	LR	0.9220	0.8858	0.9035	0.8966	0.9547
	RFC	0.9755	0.8967	0.9345	0.9312	0.9887
	NB	0.8998	0.8948	0.8973	0.8880	0.9545

5.1.2 Vectorization

The second task is about the problems of data vectorization. Because one of our independent variables "Review_clean" is in text format. So, we need to convert this variable into a numerical feature vector to perform the classification task. By applying TF-IDF, we successfully achieve this goal.

5.2 the performance of original model.

In this report, our objective is to detect whether customer recommend the products, so we focus on classification model, including linear classifier and non-linear classifier.

Firstly, we compared the fitting effect of the 4 original models to evaluate the preliminary results. It is worth mentioning that, during this process, the RECALL score is used as the evaluation metric, because it is more important for the models to identify all bad recommendation made by consumers as completely as possible, which is compliance with our project's objective.

The table below shows that the Logistic Regression and Naïve Bayes have relatively higher recall scores compared to the Random Forest and Gradient Boosting. So, it seems that the Naïve Bayes gives the best results for our analysis, as its recall score has reached nearly 0.866, followed by the Logistic Regression model, whose recall score equals to 0.862. Besides, the recall score of Random Forest is 0.846, while the score of Gradient Boosting is only 0.819.

Table 3: Performance for Original Models

Model	Precision	Recall	F1	Accuracy	AUC
LR	0.53770	0.86158	0.66215	0.84570	0.92237
NB	0.51216	0.86577	0.64359	0.83171	0.91912
RFC	0.50908	0.84648	0.63579	0.82980	0.91025
GBDT	0.46946	0.81879	0.59676	0.80580	0.88627

5.3 the performance of model after adding new variables.

In order to improve the performance of models. We explore three methods, and the first one is to add new features. According to the EDA analysis results and our past experience, we decide to add six new variables to the models and see whether the performance of models improve. However, the results of MI scores of the variables — “Have_title” and “Have_number” is 0.0067 and 0.0047, which are closed to 0, so we finally drop these two variables and only used four new variables: ‘Age’， ‘Length Review’, ‘Department Name’ , to the our models.

Before adding these new variables, we need to firstly normalize the “Age” and “Length Review” variables, because after TF-IDF vectorization, the input is a sparse matrix and the elements range is all between 0 and 1. So in order to keep the same, when we add new variables, we normalize the “Age” and “Length Review”, to keep that the all variables have the same attributes.

Then we change the “Department Name” into dummy type.

However, the final results show that these new variables don ‘t make sense in the models’ improvement, since the recall score doesn’t increase greatly, and some recall scores even decrease. For example, the recall score of Random Forest only increase from 0.86468 to 0.84899, and the scores of Logistic Regression and Naïve Bayes both decrease, from 0.86158 to 0.8599 and from 0.86577 to 0.86326, respectively.

Table 4: Performance for Models after Adding New Variables

Model	Precision	Recall	F1	Accuracy	AUC
LR_add	0.53665	0.85990	0.66086	0.84511	0.92267
NB_add	0.50565	0.86326	0.63774	0.82789	0.91448
RFC_add	0.49852	0.84899	0.62818	0.82362	0.90897
GBDT_add	0.45999	0.82466	0.59057	0.79932	0.88541

5.4 the performance of model after adjusting stop-words list.

After building GBDT model, we can get the importance of different words. The feature importance shown in TabelXX. By making statistics, we can find that it is actually important to classify the recommendation based on whether the review text contains this word or not, and sometimes it is counterintuitive. For example, we can see that it is inclined to be “bad recommendation” for group whose review text contain “wanted”.

Table5: The feature importance of words

	feature_importance		feature_importance
wanted	0.071	however	0.020
love	0.062	return	0.020
perfect	0.058	soft	0.019
comfortable	0.048	jean	0.019
great	0.046	unflattering	0.018
disappointed	0.042	perfectly	0.017
looked	0.041	unfortunately	0.016
back	0.038	huge	0.012
returned	0.027	cheap	0.012
returning	0.025	fabric	0.011
little	0.021	bought	0.011
compliment	0.021	excited	0.011
not	0.021		

Removing 'but' from stop-words list.

According to table, we try to explore the attributes of some observations whose review text contains these important words like "love", "great". We found that if "but" appears as the same time as some words like "love" in the review text, the observation is inclined to "bad recommendation". However, the word 'but' exists in the stop-words list, so we remove it.

Adding new word to stop-word list.

It's obvious that the clothes type wouldn't bring good influence on classification from the results shown in 5.3. so, in order to reduce the dimension of inputs, we add these words that describe the clothes type into stop-words list. These words include:

['intimates', 'dresses', 'pants', 'blouses','knits','outerwear','lounge', 'sweaters', 'skirts', 'fine gauge', 'sleep', 'jackets', 'swim', 'trend', 'jeans', 'legwear', 'shorts', 'layering']

New models after adjustment.

We train the four models again after the adjustment aforementioned. The performance of all models is improved. The evaluation metrics of different models are show in TableXX. We can also know that the LR and NB still have the relatively best performance.

Table6: Comparison of four models after adjustment of stop-words list.

Model	Precision	Recall	F1	Accuracy	AUC
LR_st	0.53309	0.87164	0.66157	0.84349	0.92209
NB_st	0.51767	0.87248	0.64980	0.83495	0.92170
RFC_st	0.49537	0.85319	0.62681	0.82170	0.90644
GBDT_st	0.46569	0.82550	0.59546	0.80315	0.88519

5.5 the performance of model after high-parameter tunning.

In addition to the two methods mentioned before, in order to further improve the model’s performance, our third method is to tune hyperparameter of the Random Forest and GBDT models.

For Random Forest:

There are several features that can be tuned to improve the fitting effect of the model, and among these features, we regard the “n_estimators” as the most important one, as this feature decides the number of tress we want to build before taking the maximum voting or averages of predictions. Therefore, in order to improve the coding efficiency, we only tune the “n_estimators” for the Random Forest model. When we change the ‘n_estimators’ to 160, the recall score has slightly increased from 0.846 to 0.861.

Table 7: Performance for RF Models after hyperparameter tuning

	Precision_Score	Recall_Score	F1_Score	Accuracy	AUC
RFC_best	0.501957	0.860738	0.634116	0.825677	0.912709

For Gradient Boosting Tree:

The parameters we decide to tune include “n_estimators”, “learning_rate”, “max_depth” and “min_samples_split”. After selecting the parameters, we utilize Random Search to find a better combination of these parameters.

The better combination of GBDT found by Random Search is that, the “n_estimators” should equal to 200, “learning_rate” equals to 0.1, “max_depth” equals to 5 and “min_samples_split” should be 50. Then we replace the default values with this output to see the final result. The tables below shows that the model’s performance is improved after tunning, with recall score growing from 0.819 to 0.845.

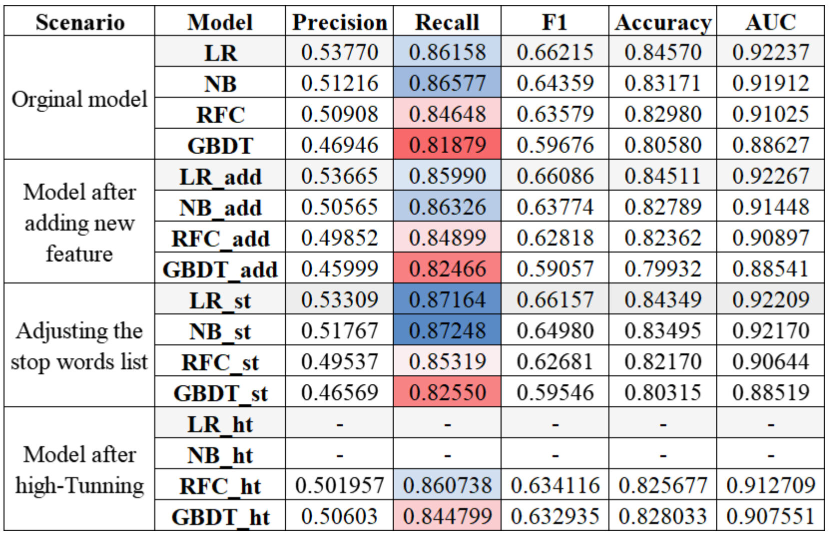
Table 8: Performance for GBDT Models after high-parameter tuning

	Precision_Score	Recall_Score	F1_Score	Accuracy	AUC
GBDT_best	0.50603	0.844799	0.632935	0.828033	0.907551

5.6 the comparison of model in all scenarios.

Overall, we trained 4 models and used 3 methods to improve the performance of models. In this part, we want to see the performance of each model in different scenarios.

Table 9: Performance in all scenarios



Scenario	Model	Precision	Recall	F1	Accuracy	AUC
Orginal model	**LR**	0.53770	0.86158	0.66215	0.84570	0.92237
	NB	0.51216	0.86577	0.64359	0.83171	0.91912
	RFC	0.50908	0.84648	0.63579	0.82980	0.91025
	GBDT	0.46946	0.81879	0.59676	0.80580	0.88627
Model after adding new feature	**LR_add**	0.53665	0.85990	0.66086	0.84511	0.92267
	NB_add	0.50565	0.86326	0.63774	0.82789	0.91448
	RFC_add	0.49852	0.84899	0.62818	0.82362	0.90897
	GBDT_add	0.45999	0.82466	0.59057	0.79932	0.88541
Adjusting the stop words list	**LR_st**	0.53309	0.87164	0.66157	0.84349	0.92209
	NB_st	0.51767	0.87248	0.64980	0.83495	0.92170
	RFC_st	0.49537	0.85319	0.62681	0.82170	0.90644
	GBDT_st	0.46569	0.82550	0.59546	0.80315	0.88519
Model after high-Tunning	**LR_ht**	-	-	-	-	-
	NB_ht	-	-	-	-	-
	RFC_ht	0.501957	0.860738	0.634116	0.825677	0.912709
	GBDT_ht	0.50603	0.844799	0.632935	0.828033	0.907551

5.6.1 Logistic Regression

Logistic Regression Model has shown good performance, which has a high recall score in original model of 0.862. Although adding new features shows little contribution to the model, after we adjusted the stop words list, its recall score increased to 0.872.

5.6.2 Naive Bayes

Naive Bayes has the best performance among models. Its recall score is 0.867 in the original model, after adjusting new features, the recall score decreases slightly to 0.863, but increases to 0.872 after stop words list adjustment.

5.6.3 Random Forest

With a recall score of 0.846 in the original model, we find that the model performance becomes better when new features added, which increases to 0.849. After adjusting the stop word list, its recall score continues to increase to 0.853. Considering "n_estimator" as the most important hyperparameter, we tune it for Random Forest Model, and finally get the recall score of 0.861.

5.6.4 Gradient Boosting Decision Tree

In the original model, GBDT has a relatively low recall score of 0.819. After adding new features and adjusting stop word list, its recall score gradually increases to 0.826. Utilizing Random Search method to conduct hyperparameter tuning, we get a better model performance for GBDT, with a recall score of 0.845.

6. Conclusion

6.1 the model-level conclusion

The first conclusion is from model-level perspective. According to the performance matrix and modeling analysis mentioned above, we can see that Naive Bayes and Logistic Regression give the best results for our topic, meaning that these two models are effective at predicting recommendations. Besides, Naive Bayes is more time efficient, which is important when we solve with a large data set.

6.2 the business application

The second one is about the business application. From business theory's standpoint, recommendation can represent the consumer's loyalty which means the extent to consumer's continued purchasing behavior in future. Meanwhile, in the age of online media, piggybacking consumer's recommendation is crucial marketing method. So, in both cases, it's important to identify the consumers' recommendation situation.

When there is no "recommendation" indicator, we can detect the consumers without enough satisfaction by our models. By looking through their reviews, With a high recall score, in the absence of a "recommendation" indicator, our models can identify consumers who give bad recommendations better. By tracing the reviews of such consumers, we can know their feelings and opinions about products or service. These opinions may can lead online shops' to find problems with its products or services, and then to further improve consumers' satisfaction.

When we identify consumers with "bad" recommendation, we can also implement remedial measures. Remedial measures can include:

(1) Email or call back to ask why they are unsatisfied;
(2) Give a sincere apology and some coupons to these customers;
(3) Conduct relevant staff training