Dataset Overview

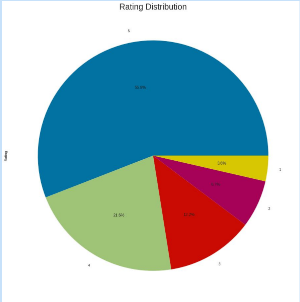
- Our dataset includes 23486 rows and 10 feature variables
- Each row includes a customer review
- In our analysis we mainly focused on 3 variables: Review Text, Rating and Recommend_ind

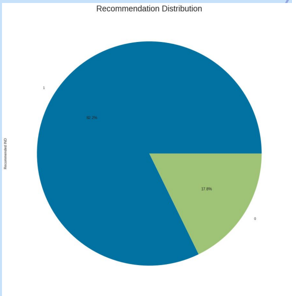
Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses
1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses
1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants
847	47	Flattering shirt	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses

Goal:

Using review text to:

- predict rating on a 5-point scale
- predict whether the customer would recommend the product to others



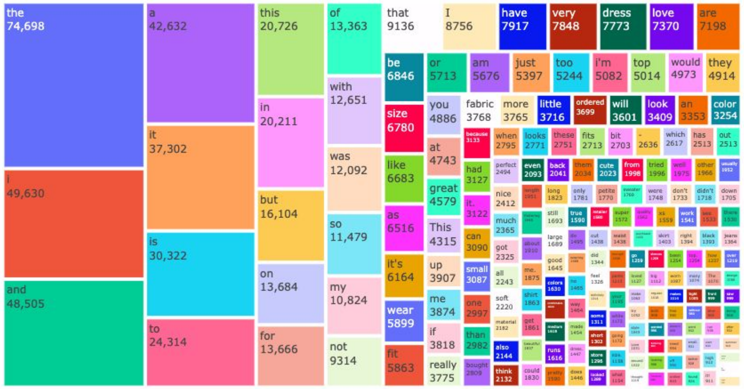


Preprocessing

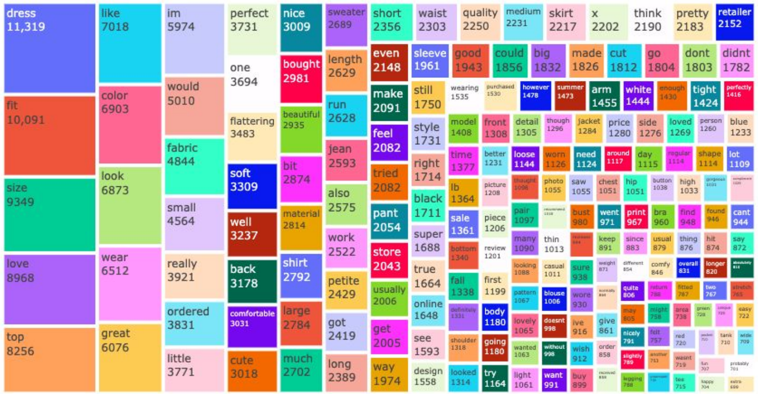
Review Text:

1. Remove punctuations and numbers
2. Tokenize- split the string into smaller units
3. Removing Stopwords- ‘it’, ‘the’, ‘a’ etc.
4. Lemma- aiming to establish structured semantic relationships between words
5. Joining cleaned text back together

Top Frequent 200 Words in the Dataset (Before Cleaning)



Top Frequent 200 Words in the Dataset (After Cleaning)



Preprocessing

Handling rare words/ frequent words:

Two methods of vectorization

- `TfidfVectorizer(max_df=0.8, min_df = 5)`
- `CountVectorizer(max_df=0.8, min_df=3)`

Before preprocessing:

```
1 .  Absolutely wonderful - silky and sexy and comfortable

4 .  I love, love, love this jumpsuit. it's fun, flirty, and fabulous! every time i wear it, i get nothing but great compliments!

12 .  This dress is perfection! so pretty and flattering.

14 .  Bought the black xs to go under the larkspur midi dress because they didn't bother lining the skirt portion (grrrrrrrrrrr).
my stats are 34a-28/29-36 and the xs fit very smoothly around the chest and was flowy around my lower half, so i would say it's running big.
the straps are very pretty and it could easily be nightwear too.
i'm 5'6" and it came to just below my knees.
```

After preprocessing:

```
1 .  absolutely wonderful silky sexy comfortable

4 .  love love love jumpsuit fun flirty fabulous every time wear get nothing great compliment

12 .  dress perfection pretty flattering

14 .  bought black x go midi dress didnt bother lining skirt portion stats x fit smoothly around chest flowy around lower half would say running big strap pretty could easily im came knee
```

Count/ TF-IDF Vectorization

Set up

- Want to predict if the customer recommends the product or not based on a review text
    - Train/Test split
        - With the test size proportion being 0.2
        - Stratifying based on y (Recommended IND or Rating)
    - Construct a pipeline, within which
        - do the TF-IDF / Count vectorization on the review text
        - With max_df = 0.8, min_df = 5
        - Call a classification model
    - Search for the best hyperparameters, including
        - The ngram_range for TF- IDF Vectorizer
        - Hyperparameters for classification models themselves
    - Finally, fit the best model

- **Model considered:**
    - Logistic Regression l1 penalty
    - Gradient Boosting Classifier
    - Random Forest Classifier
    - etc.

```
pipe_lr = Pipeline([('tfidf',TfidfVectorizer(max_df=.8, min_df = 5)),
                    ('lr', LogisticRegression(penalty = 'l1',
                                              solver = 'liblinear',
                                              random_state = 123))])

params = {'tfidf__ngram_range':[(1,1), (1,2), (2,2)],
          'lr__C':[0.1,1,10,100]}
gscv = GridSearchCV(pipe_lr, params, cv = 3, n_jobs = -1).fit(X_train_r, y_train_r)
```

Logistic regression TF-IDF example

Recommendation: TF-IDF Vectorization

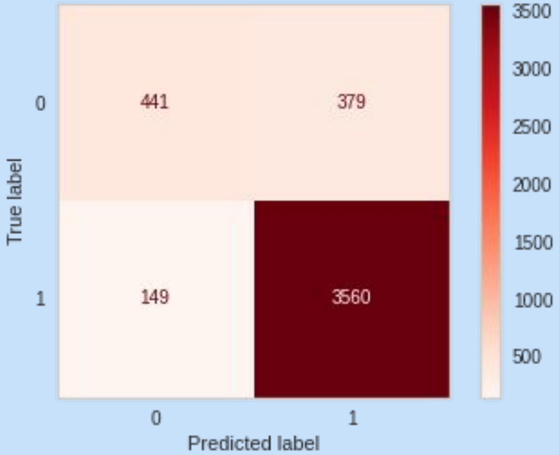
Model Selection

Model Name	Best ngram_range	Best Model Hyperparameters	Best Model Training Score	Best Model Testing Score
Logistic Regression	(1, 2) (From (1, 1), (1, 2) and (2, 2))	C = 1 (From 0.1, 1, 10 and 100)	0.9033	**0.8834**
Gradient Boosting Classifier	(1, 2) (From (1, 1), (1, 2) and (2, 2))	learning_rate = 0.3 (From 0.1, 0.2, 0.3, 0.4 and 0.5)	0.9166	0.8699
Random Forest Classifier	(1, 2) (From (1, 1), (1, 2) and (2, 2))	max_depth = 90 (From 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100)	0.9764	0.8488

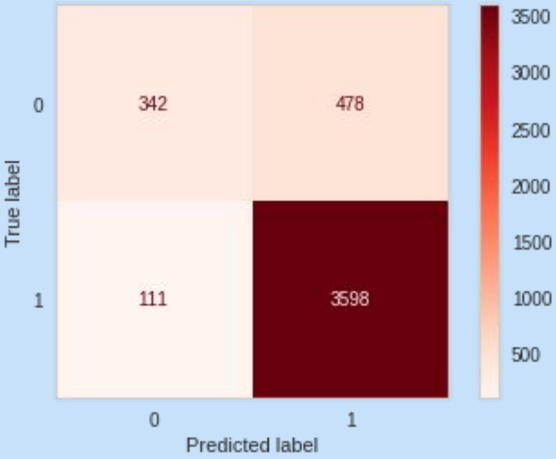
Recommendation: TF-IDF Vectorization

Confusion Matrix

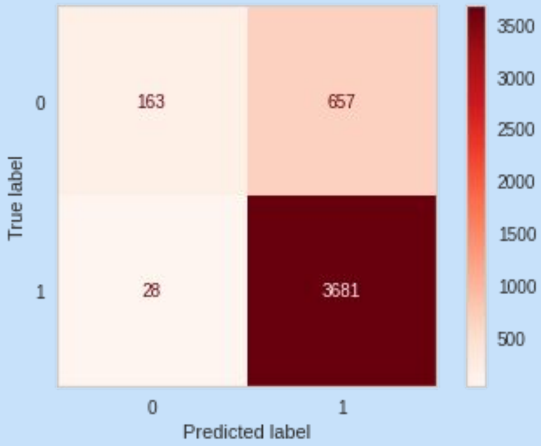
Logistic Regression



Gradient Boosting Classifier



Random Forest Classifier



- Good at predicting “Recommended”
- Not Good at predicting “Not Recommended”
    - Some models such as Gradient Boosting Classifier and Random Forest Classifier even predict “Not Recommended” as “Recommended” more than “Not Recommended”

Rating: TF-IDF Vectorization

Model Selection

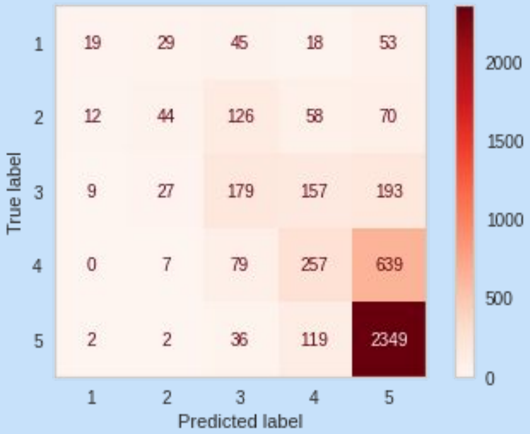
Model Name	Best ngram_range	Best Model Hyperparameters	Best Model Training Score	Best Model Testing Score
Logistic Regression	(1, 2) (From (1, 1), (1, 2) and (2, 2))	C = 1 (From 0.1, 1, 10 and 100)	0.6714	**0.6288**
Gradient Boosting Classifier	(1, 2) (From (1, 1), (1, 2) and (2, 2))	learning_rate = 0.2 (From 0.1, 0.2 and 0.3)	0.7418	0.6039
Random Forest Classifier	(1, 1) (From (1, 1), (1, 2) and (2, 2))	max_depth = 90 (From 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100)	0.9899 (potentially overfitting)	0.5816

- Models' performances on rating prediction is worse than recommendation prediction

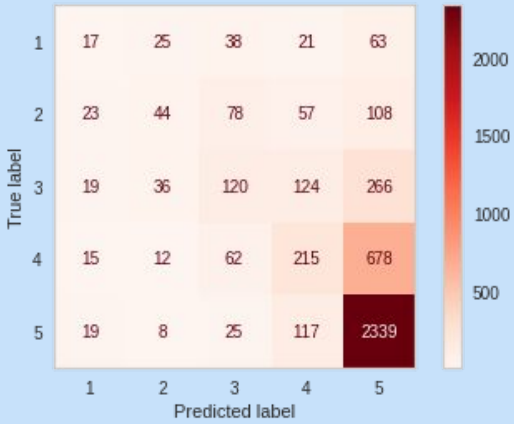
Rating: TF-IDF Vectorization

Confusion Matrix

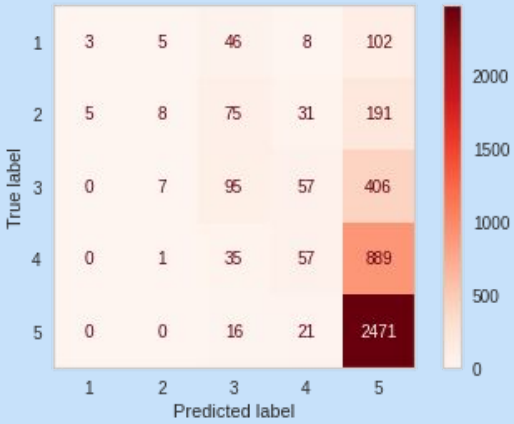
Logistic Regression



Gradient Boosting Classifier



Random Forest Classifier



- Good at predicting rating score 5
- Not Good at predicting all other rating scores
    - The models tend to predict a rating score other than 5 to a higher score than what it truly is
    - Particularly, Random Forest Classifier tends to predict all scores as 5.

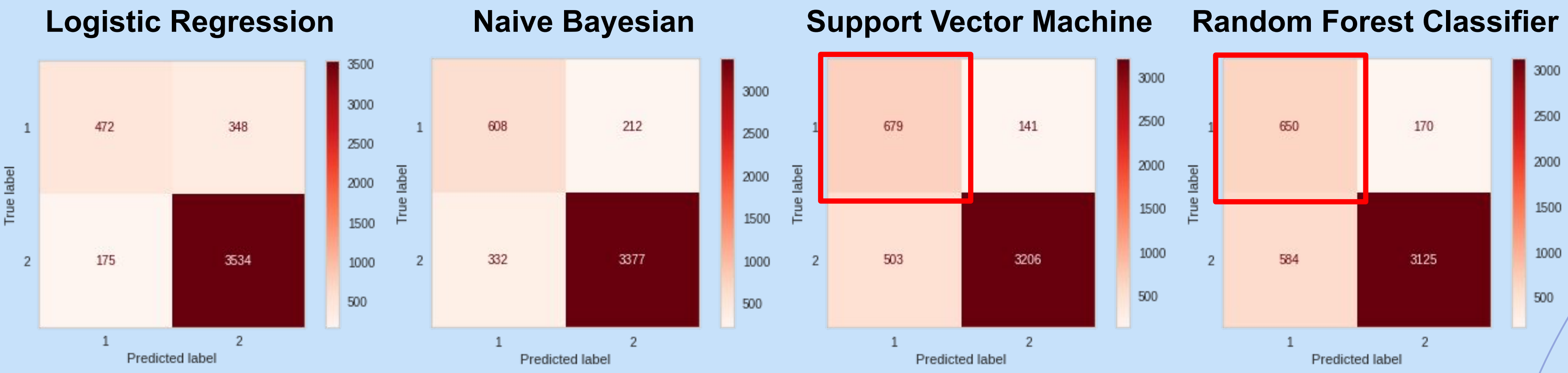
Recommendation: Count Vectorization

Model Selection

Model Name	Best Model Hyperparameters	Best Model Training Score	Best Model Testing Score	Best Model F1-score
Logistic Regression	'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'	0.9130	**0.8845**	**0.9311**
Naive Bayesian	Default	0.9034	0.8799	0.9255
Support Vector Machine	C=0.01, class_weight="balanced"	0.8968	0.8578	0.9087
Random Forest Classifier	'max_depth': 6, 'n_estimators': 11	0.8527	0.8335	0.8923

Recommendation: Count Vectorization

Confusion Matrix



- Good at predicting “Recommended”
- Not Good at predicting “Not Recommended”
    - Support Vector Machine and Random Forest Classifier perform better at predicting “Not Recommended”.

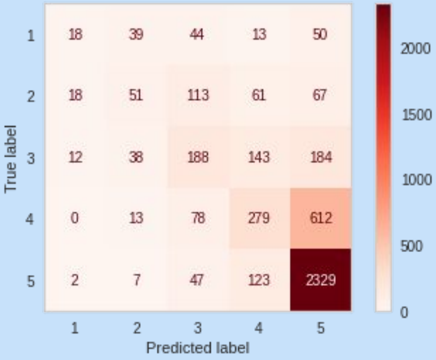
Rating: Count Vectorization
Model Selection

Model Name	Best Model Hyperparameters	Best Model Training Score	Best Model Testing Score
Logistic Regression	'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'	0.7160	**0.6326**
Naive Bayesian	Default	0.7250	0.6235
Support Vector Machine	C=0.01, class_weight="balanced"	0.7416	0.6200
Random Forest Classifier	'max_depth': 9, 'n_estimators': 11	0.6635	0.5745

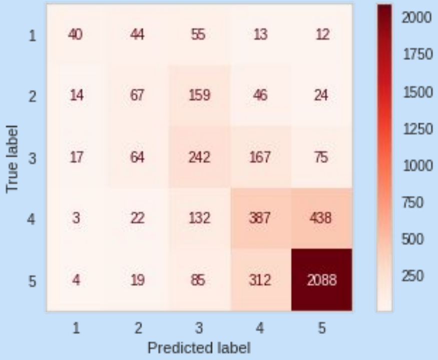
Recommendation: Count Vectorization

Confusion Matrix

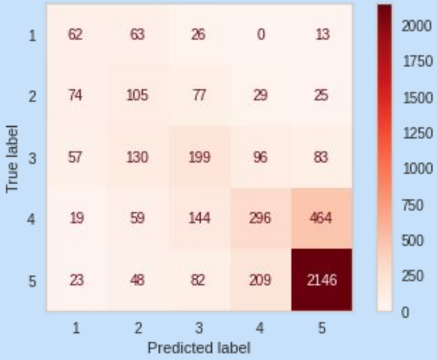
Logistic Regression



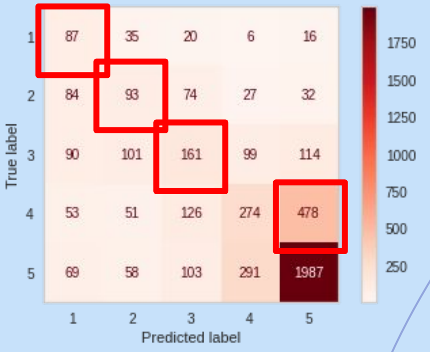
Naive Bayesian



Support Vector Machine



Random Forest Classifier



- Similarly, good at predicting rating score 5.
- Not Good at predicting all the other rating scores:
  - The models tend to predict the rating other than 5 to be higher than what it truly is.

LDA

- Want to explore the potential reason that
  - Models predict higher ratings and more recommendations

- For each of the recommendation and rating cases:
  - First, do the TF-IDF vectorization on the review text
    - With max_df = 0.8, min_df = 5, ngram_range = (1,2)
  - Then, apply the Latent Dirichlet Allocation on the TF-IDF transformed review text
    - n_components = 2 for the recommendation case since we only have 2 classes
    - n_components = 5 for the rating case since we have 5 scores
  - After getting feature names and store it, use np.argsort to select the top 20 words for each topic

```
tfidf = TfidfVectorizer(max_df=.8, ngram_range = (1,2), min_df = 5)
X_tfidf = tfidf.fit_transform(df["review_text"])
lda = LatentDirichletAllocation(n_components=2, n_jobs = -1, random_state=123) #two topics
X_lda = lda.fit_transform(X_tfidf)
vocab = tfidf.get_feature_names()
```

Two topics example−TF-IDF and LDA

```
vocab = [items.replace(' ', '_') for items in vocab]

vocab = np.array(vocab)

for ind in range(lda.components_.shape[0]):
    index = np.argsort(lda.components_[ind])[::-1][:20]
    terms = vocab[index]
    terms = ' '.join(terms)
    print(f'Topic #{ind:2d} : {terms}')
```

Two topics example−sorting the top 20

LDA— Recommendation Case

- The top 20 words for each of the two topics are:

```
Topic # 0 : size top fit love small im wear great dress run ordered little large cute shirt color medium bought perfect would
Topic # 1 : dress look like sweater color love great fabric beautiful really would much back fit make wear soft feel material work
```

	Potentially positive words	Potentially Negative words
First Topic	love, great, cute, perfect	small, large
Second Topic	like, love, great, beautiful, fit (maybe?), soft	back (maybe?)

- There are more positive words than negative in each topic
- Indicate more frequent appearance of positive words in people's reviews

LDA— Rating Case

- The top 20 words for each of the five topics are:

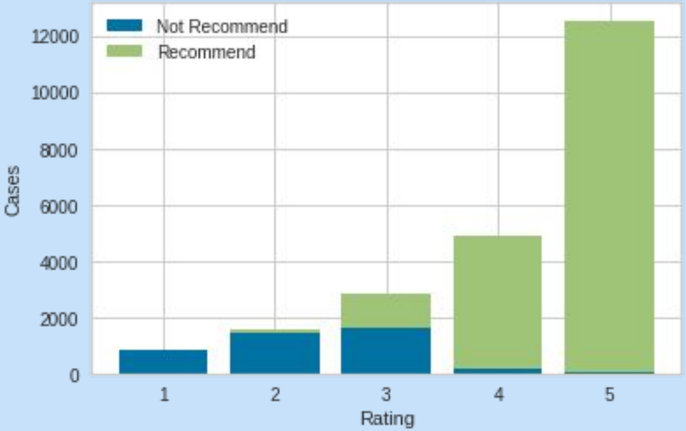
```
Topic # 0 : size dress fit top love im wear small great color ordered would run perfect petite large like flattering cute one
Topic # 1 : dress look love color like beautiful great fit make cant fabric good picture comfy loved looked back even top boxy
Topic # 2 : dress top look fabric like love fit color size wear would much great flattering im really nice skirt little one
Topic # 3 : like sweater size fit small color im look would top wear love ordered dress really large medium shirt one back
Topic # 4 : great love jean comfortable fit perfect pant soft super wear dress color pair cute look flattering compliment size top comfy
```

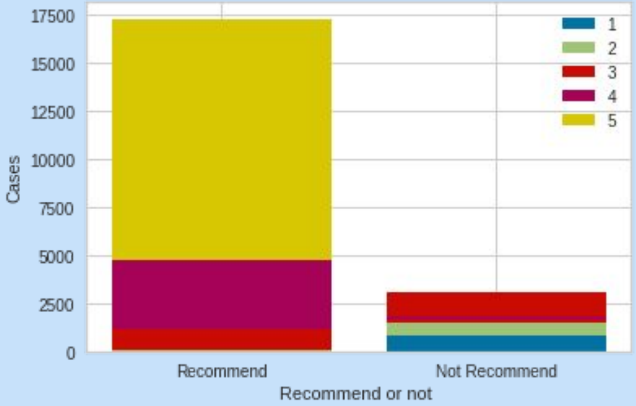
	Potential Positive Words	Potential Negative Words
First Topic	fit (maybe?), love, great, perfect, like, flattering, cute	small, large
Second Topic	love, like, beautiful, great, fit (maybe?), good, comgy, loved	back (maybe?)
Third Topic	like, love, fit (maybe?), great, flattering, nice	little (maybe?)
Fourth Topic	like, fit (maybe?), love	small, large, back (maybe?)
Fifth Topic	great, love, comfortable, fit (maybe?), perfect, soft, cute, flattering, compliment, comfy	NA

- There are more positive words than negative in each topic on average
- We think this can be one situation explaining more frequent appearance of positive words in people's reviews and therefore make models predict more positively: some people may tend to write some good points about the product even though their overall attitude is negative, and this confused our models.

Why Models Perform Better on Predicting Recommendation?

- Want to explore the potential reason why our models perform better on predicting recommendation than rating

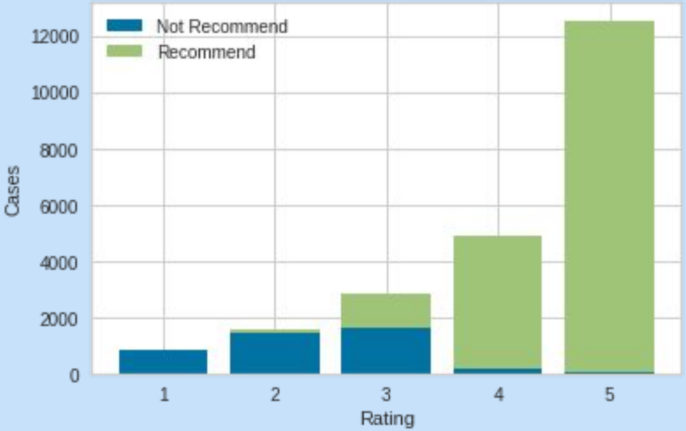


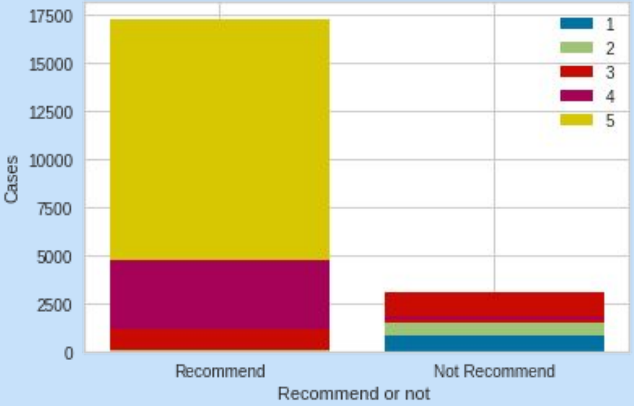


- The stack plot on the left shows that not only people who give 5 star recommend (almost all people giving 5 star recommend), but also people who give 3 and 4 stars also recommend (about half of people giving 3 star recommend, and almost all people giving 4 star recommend)
- The stack plot on the left shows that people who recommend are far more than who do not

Why Models Perform Better on Predicting Recommendation? — Continued

- Want to explore the potential reason of why our models perform better on predicting recommendation than rating





- While the more frequent appearance of positive words in people's reviews can mislead the models to predict a higher rating, such impact decreases in recommendation case since people who give 3, 4, and 5 star can potentially recommend the product, which decreases the need for models to be as sensitive as in the rating classification case

- This may be why models perform better on predicting recommendation than rating

Summary & Future Work

Summary:

- Used review text of product to predict both product rating on a 5-point scale and product recommendation
- While rating prediction test score settled around 0.63, recommendation achieved 0.88 of test score
- Using LDA to explore model bias

Future work:

- Explore combination of both text review as well as other features (item department, customer age etc.) in a predictive model
- Filter more neutral words (e.g.product names) out to focus only on words associated with customers' attitude
- Determine whether the word is negative, positive or neutral first, then train to model to predict the ratings

Reference

- Data:

Women’s E-Commerce Clothing Reviews. *Kaggle*, https://www.kaggle.com/datasets/nicapotato/womens-ecommerce-clothing-reviews. Accessed 26 April 2022.

- Code: https://colab.research.google.com/drive/1-YbFJcGhTE1g9exF0u48udyZr-kpVe_y?authuser=1#scrollTo=DnP8w3JQKTQR

Thank You