Amazon Clothes Rating Prediction based on Customers' Reviews

CSE 158 Assignment 2

ZhiMin Lin
A13801142
zhl297@ucsd.edu
UC San Diego
San Diego, CA

ZhaoKai Xu
A14738474
zhx121@ucsd.edu
UC San Diego
San Diego, CA

ZiChao Wu
A13645764
ziw255@ucsd.edu
UC San Diego
San Diego, CA

ABSTRACT:

In this project, we built a model to predicts the rating of an item review given by a user. The features including TF-IDF of review text, item price, and item category are observed and evaluated. We utilized the Amazon Clothes Record dataset provided on Kaggle, which contains 200,000+ reviews. This dataset provides the necessary information to accomplish our goal of rating prediction. To obtain a high accuracy in prediction, we manipulated and mined the dataset by multiple models we learned from CSE 158 and the further reading materials. We applied logistic regression (linear model), linear SVC (linear model), KNN (non-linear model), Naïve Bayes (statistic), Latent Factor Model to build different models. And we finally obtained our best model by utilizing logistic regression, which helps to predict the rating of an item that is highly possible to be assigned by a user, and thus helps the sellers to better promote their products to a targeted user effectively.

Keywords: TF-IDF, Amazon Clothes dataset, reviews, logistic regression, linear SVC, KNN, Naïve Bayes, Latent Factor Model

INTRODUCTION:

As shopping clothes online is growing more and more popular (based on the plotted diagram of reviews growing from 2003 to 2014), having a better understanding of the unique preference of each user will be crucially beneficial in improving the online clothes market on Amazon. Meanwhile, by promoting a clothes to a user that he/she might prefer can significantly improve a his/her satisfaction and thus change his/her shopping habits. Therefore, to build a good clothes recommended system is a win-win opportunity for both users and sellers.

Based on the dataset collected from Amazon, by applying data analysis and data mining, we can better understand customers' online shopping preference. And by applying the recommended system, we can promote the item to specific user groups effectively.

In this paper, our main task is to analyze user's reviews on clothes item and predict their rating on the product based on Amazon clothes dataset. Besides, we applied some other relevant features such as price, categories, and helpful rate of a review that may affect the prediction as filters. Our first step is to plot some histograms to analyze data so that we can have an overview on data distribution alongside important features. As we defined our task as a classification problem to predict the item rating given a (user, item) pair, we trained data on multiple classifiers including linear, non-linear, and statistical models to make the rating prediction. Then we optimize our models by modifying parameters with the help of confusion matrix, MSE plot, and other diagrams. Finally, we promoted the items to the user if and only if the item achieved a high predicted rating.

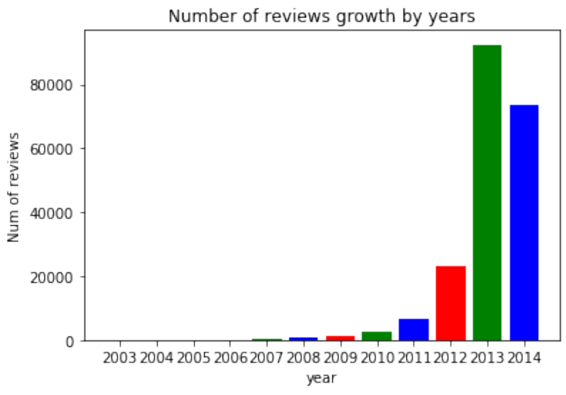


Figure 1: Amazon Reviews Growth by years

1) DATA

1.1 Dataset Properties:

The Amazon clothes dataset from Kaggle consists reviews from 2003 to 2014, including 200,000 reviews from true users. Each review contains a list of information including category description, category ID, rate of viewers who think the review is helpful, item ID, item price, review rating, review hash code, review plain Text, review time, reviewer ID, review summary, and Review Time. Based on this dataset, we ran an exploratory analysis to explore the data distribution and interesting findings.

Properties	Data Types	Data Sizes
Reviews Text	String	200,000
Review Rating	Int (1 – 5)	200,000
Items ID	String	19,914
User ID	String	39,239
Review Price	Float	74,111

Table 1: Dataset Properties and Type

According to our prediction task, we were the interested in exploring the features that mostly affect review rating. Also, review feedback rated by other viewers can potentially indicate

Figure 2: Amazon Clothes Items Diversity

1.2 Exploratory Analysis (Statistic)

To better understand the dataset and extract non-trivial features, we performed an exploratory analysis on the data. By applying the statistical analysis, we tried to plot data distribution alongside different features (rating, price, category, popularity, helpful feedback) that we may consider use in our models.

1.2.1 Review Ratings

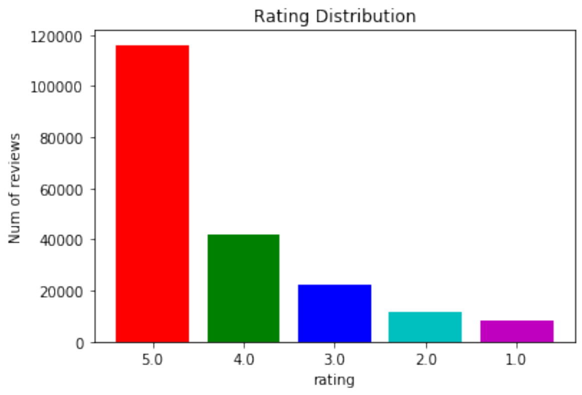


Figure 3: Amazon Review Rating Distribution

From the dataset collecting reviews in recent ten years, we can see 158,281 users left positive reviews (rating > 4.0), which takes 79.14%. By setting a proper threshold, we should be able to filter out some low rating items that will never be recommended to a customer.

1.2.2 Clothes Prices

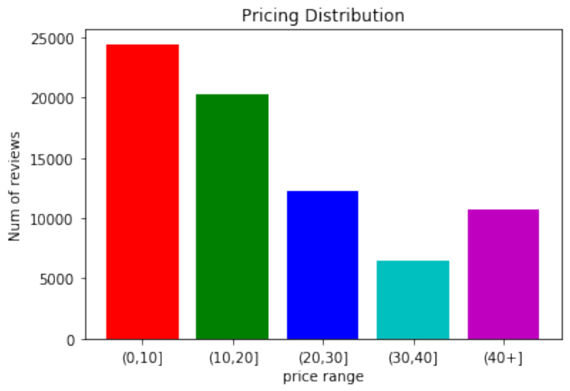


Figure 4: Amazon Clothes Item Price Distribution

From the 200,000 reviews, we found there are totally 74,111 unique price items, and the other 2/3 of the items that do not have price. Over 60% of the items are under the price of 20 dollars. So we can draw a rough conclusion that the price distribution is not well balanced and most of the items are comparably cheap.

1.2.3 Clothes Categories

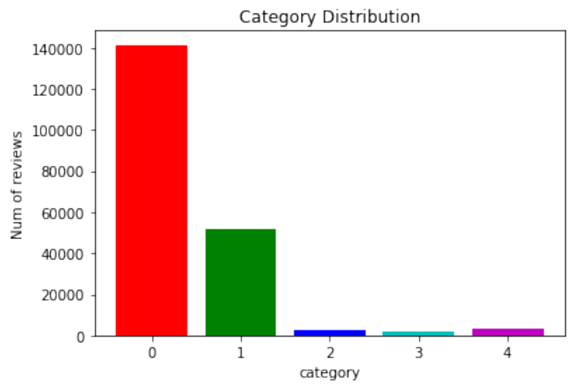


Figure 5: Amazon Category Distribution

From the clothes categories distribution histogram, we found that 141,398 reviewed items are category 0, which takes 70.69% of the entire dataset. From the very uneven data distribution, we came with an intuitive thought to apply different classifiers under different categories, and target our classifiers mainly on category zero and one that have a significant number of reviewers.

1.2.4 Clothes Popularity

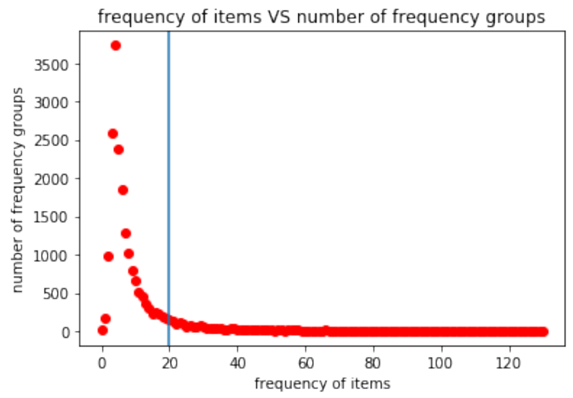


Figure 6: Item Popularity Distribution

We performed a statistical calculation based on the frequency of an item that is been purchased and plot a diagram to present the item popularity. Due to the highly unbalanced data distribution of item popularity, we realized that most items are aggregated in frequency range 0 to 20. Therefore, by setting a threshold we can effectively filter out some low frequency items.

1.2.5 Review Recommended by Other Users

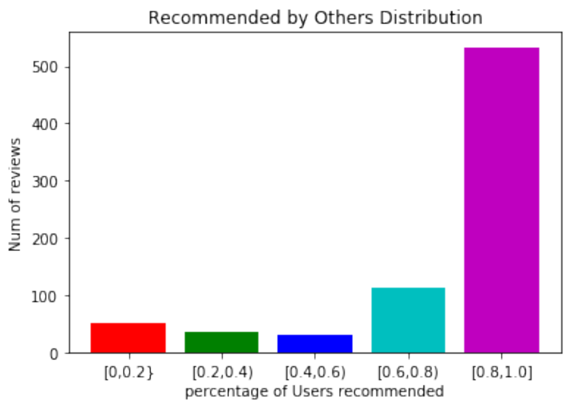


Figure 7: helpful review percentage

To better understand the quality of reviews ranked by other users, we filtered out all the reviews that do not have any helpful record and calculated helpful votes percentage by taking the division: (helpful number) / (overall votes). From the histogram, we can notice that users are more preferred to recommend an item and thumbs up for a review if he or she likes the item. As a result, the items that are associated with helpful reviews (recommended >0.8) can be highly recommended to a user.

1.3 Review Texts

The review text is the most crucial feature that we can take advantage of in making rating prediction because it contains many keywords to indicate a user's preference.

From the review text, we collected 100 most frequent words and presented an intuitive visualization with the help of word cloud library.

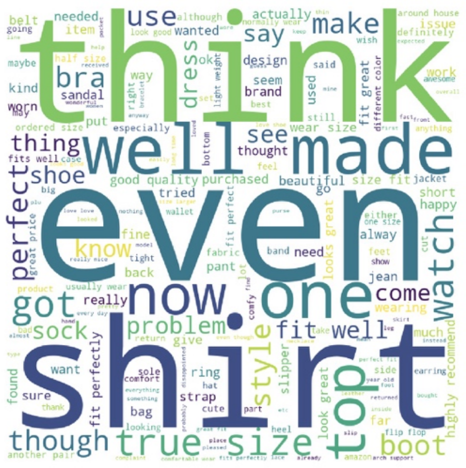


Figure 8: popular keywords in positive reviews

1.4 Interesting Findings

Based on statistically analysis, we found an interesting thing is the reviews are more centralized within a range alongside a feature. So by paying more attention in build the model using those centralized features, it would be more likely to obtain a better result.

2) PREDICTIVE TASK

We defined the problem to e a classification problem and our task to be predicting the review rating score from 1.0 to 5.0. The models that we implemented are evaluated based on prediction accuracy and (MSE) mean square errors. We separated the overall dataset into training, validation, and testing while performing our experiment. To make an accurate prediction in testing, we trained and evaluated our models on the training and validation dataset, reported and visualized the performance of different parameters in tuning to achieve the best parameters for a model (while the performance are both relatively good in training and validation), and reported the model performance in testing.

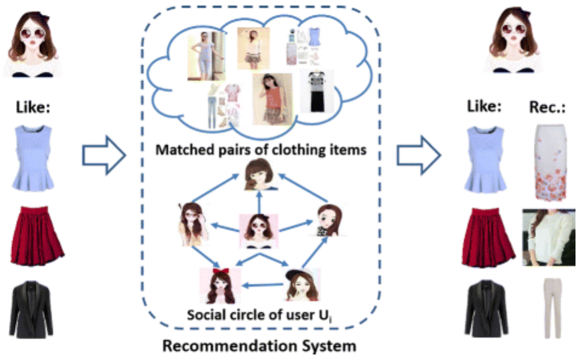


Figure 9: Amazon Clothes recommendation system

2.1 Evaluation Methods

Two simple baseline models are used to study the dataset: average rating and Dummy Classifier from sklearn library.

For every model in the study, we trained the model based on 50% of the dataset for we calculated its performance on the test set which is 50% size of the entire dataset (20,000 entries).

The result of the prediction is evaluated by two factors: accuracy and mean squared error. The overall accuracy is calculated as the number of overall correct predictions divided by the size of the test set. The MSE is calculated by the following function:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Logistics Regression, Naïve Bayes, linear SVC and KNN was evaluated by the accuracy, since they all classifies the prediction to one of the integer ratings in range [1.0, 5.0]. The Latent factor mode, on the other hand, is difficult to evaluate by the accuracy, since the output is real value data. Therefore, we use MSE to measure the performance instead and compare it to the baseline.

2.2 Data Preprocessing and Feature Selection

2.2.1 randomly shuffle the data

We shuffled the 200,000 dataset by calling the built-in library and then separated dataset into training-validation and test dataset.

2.2.2 Bag of words (on review text)

We firstly extracted all the unigrams from all the review texts. We then counted the frequency of each unique unigram and sort them in descending order according to their frequencies. After the sorting, we took the top 1,000 most frequent words to build our feature vectors for each review text.

For each review text, we created a feature vector of size 1000. We counted the frequency of each most frequent word in this text, and put this frequency into the corresponding position in the feature vector.

2.2.3 TF-IDF (on review text)

TF-IDF is a numerical measurement of the importance of a word in a collection of documents. TF represents term frequency and IDF represents inverse document frequency

TF

$tf(t,d)$ = number of times the term t appears in the document d

e.g. TF ("jacket", "Jackie e is wearing a great jacket, and I like his jacket") = 2

IDF

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

TFIDF

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

High TF-IDF means this word is frequently appeared in this document compared to other documents, while low TF-IDF represents this word appears infrequently in this document, or it appears in many documents.

So, the features to feed into our models are bags of words and TF-IDF generated from review texts.

3) MODELS AND METHODOLOGY

3.1 Baseline

We chose two base line models. The first one is to naively predict all the ratings to be the average rating of the training set. The average rating of the training set is 4.237.

The second model is to use a Dummy Classifier in the sklearn library. This classifier basically generates predictions by respecting the training set's class distribution. For example, if 40% of the review have a rating of 5.0, then there are 40% chance for a predicted rating to be 5.0. This is a naïve classifier using only basic statistics.

The input to the Dummy Classifier is almost unprocessed text strings, except that punctuations have been removed. Each string is a review text from the dataset.

3.2 Naïve Bayes

Naïve Bayes model make predictions based on computing the probability of a feature appear with a given label. In this practice, we chose the Gaussian Naïve Bayes model to train and predict, assuming that the distribution of the features are close to Gaussian. The probability is given by the following equation:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

where x_i is one of the features and y is a label.

The advantage and disadvantage of Naïve Bayes model are both clear. The advantage is that the model is relatively simple and only uses statistical method. Therefore, the model can be trained with a relatively low cost of time, and does not require tuning to any parameters (versus the adjustment for regularizer in supervised models).

The disadvantage, on the other hand, is also obvious. The model relies totally on statistics, and therefore does not perform any learning process. This prevent it from utilizing any underlying features and pattern, which could be a limitation to its accuracy. Also, we make assumption to the distribution of the features, which might not always be the best case.

3.3 Linear SVC

Linear support vector classifier is a commonly used supervised model for text classification. It focuses on minimizing the number of misclassifications:

$$\mathrm{arg}min_\theta \sum_i \delta(y_i(X_i \cdot \theta - \alpha) \leq 0$$

Linear SVC is chosen here mainly because it is capable of performing supervised learning and prediction effectively on high-dimensional spaces (i.e. features). With the feature vector having a size of 1000 and a training set of size 10000, linear SVC can still run effectively without taking too much time.

In practice, we used the linearSVC classifier in the sklearn library to perform the predictive task. Since we have 5 classes (integer rating from 1.0 to 5.0) to predict, linearSVC is an ideal choice since it implements a "one-vs-the-rest" multi-class strategy, which trains n_class models and thus is much faster than the other general SVC models in sklearn library that trains n_class $*$ (n_class $-$ 1) / 2 "one-vs-one" models.

This model is run with both words of bags and TF-IDF feature vectors.

In terms of tuning the regularizer value, we trained five classifiers using [0.01, 0.1, 1, 10, 100] for the value of the regularizer, and choose the one with highest accuracy on the val set.

The strength of linear SVC is very efficient in runtime and performs comparably high accuracy for classification tasks. However, it can only be fed in with feature representations like TF-IDF and bags of words, But it can not be fed in with structural data considering the order of words.

3.4 Logistic Regression

Linear support vector classifier is another commonly used supervised model for text classification. Similar to Naïve Bayes, the model still aims to predict p (label | features) , but instead of directly counting, trains a classifier to prevent double counting problem resulted from treating features independently and then uses the sigmoid function

$$\mathrm{f(x)} = \frac{1}{1+e^{-x}}$$

to convert the predicted real value into a probability in the interval [0, 1].

The likelihood function is:

$$\mathrm{L} = \prod_{y_i=1} p_\theta(y_i|X_i) \prod_{y_i=0} (1 - p_\theta(y_i|X_i))$$

Taking logarithm on both sides yields the log-likelihood:

$$L_\theta(\mathrm{y}|\mathrm{X}) = \sum_i -\log\left(1+e^{-X_i\cdot\theta}\right) + \sum_{y_i=0} -X_i \cdot \theta$$

Adding a penalty term makes the function become

$$L_\theta(\mathrm{y}|\mathrm{X}) = \sum_i -\log\left(1+e^{-X_i\cdot\theta}\right) + \sum_{y_i=0} -X_i \cdot \theta - \lambda||\theta||_2^2$$

We used the LogisticRegression model in the sklearn library to perform the prediction task in practice. Since logistic regression itself is a binary classifier, the training algorithm of the model uses the "one-vs-rest" scheme to train n_class (5 in this case) models to build up the classifier.

This model is run with both words of bags and TF-IDF feature vectors.

In terms of tuning the regularizer value, we trained five classifiers using [0.01, 0.1, 1, 10, 100] for the value of the regularizer, and choose the one with highest accuracy on the validation set. This is the same as we did with the linear SVC model.

The strength of logistic regression is fast in computing and can achieve comparably high accuracy, while its weakness is it doesn't perform well when feature space is too large, and it doesn't handle large number of categorical features/variables well.

3.5 K Nearest Neighbor (KNN)

KNN is used to classify which rating category will a product fall into, based on its TF-IDF transformed review text. KNN rating predictions directly reflect the similarity of item review texts. As a nonlinear classifier, KNN is good for predictions in complex feature space distribution.

Steps:

```
1: procedure K-NN(a, b)
    Initialize k random centroids
2: while cluster assignments change between iterations do
3:  Assign each Xi to its nearest centroid
4: Update each centroid to be the center of points assigned to it
```

In the KNN model, we ran into memory issue due to the large scale of dataset. Since KNN stores the 200,000 dataset in high feature dimensions, the memory cost is comparably large and personal computer can not perform the calculation effectively.

The advantage of KNN is it does not rely on any assumption about data, and it is very simple algorithm to implement and interpret. But it requires very large scale of memory issues, which means is expensive in computing.

3.6 Latent Factor Model

The simple latent factor model after pre-filtered the training data

Rating = $\alpha + \beta user + \beta item + \gamma user * \gamma item$

where

$\alpha -$ Global average rating

$\beta user$ –Inclination of user to give better rating

$\beta item$: Inclination of item to receive better rating

$\gamma user$*$\gamma item$ – The latent factor term.

A simple MSE regularizer is used for hyper-parameter tuning. The formula is given as follows:

Regularizer = λ*[|$\beta user$|^2+ |$\beta item$|^2] + |$\gamma user$ |^2 +|$\gamma item$|^2

The model was trained on the train set. We used vanilla gradient descent model for optimization.

The value spit out by the final latent model is a decimal value for the rating. Since we're making a categorical prediction, we have to round off the predicted value to the nearest rating, which is an integer. If predicted = 4.3, we round the prediction to be a 4, etc.

4) LITERATURE

4.1 Related Works

Our dataset was collected by McAuley and used by him and He when they proposed a model with one-class collaborative filtering to predict the evolution of fashion trend using image data. Instead of using user ratings and review to predict users' fashion dynamics, the paper explored using users' purchase history and the image of the items purchased to produce a ranking of items that has not been purchased. This is much beyond the problem that we focus on in this project, and in fact, it is in a totally different dimension of research from the one we are in. In this assignment, our objective is to predict the user's rating based on the review text that the user gives, which is categorized as the problem of sentiment analysis.

Another paper that are more related to our task here is written by Joulin, Grave, Bojanowski and Mikolov from Facebook AI Research. The paper discussed various techniques for text classification, including bag of words, hierarchical softmax, n-gram features, and convolutional model.

4.2 Relevant dataset

The dataset that we use is only a small portion of the complete dataset collected by McAuley. There are other categories such as books and Sports and Outdoors. There are also a lot of datasets on Kaggle that are similar to the one that we use, such as review datasets for fine food, electronics product and video games.

The most common ways of sentiment analysis and text classification include SVM, logistic regression and KNN. TF-IDF is widely used to extract deeper features out from the corpus as a processing method, while Bag-of-Words, one of the simplest technique, also serves as an important way of building features for classifiers. There are also more advanced approaches that use neural network to perform machine learning on the dataset.

5) RESULTS AND CONCLUSION

5.1 Baseline

Predicting all the ratings to be the average rating of the training set yields a MSE of 1.2455 with the average being 4.237.

Using the DummyClassifier from sklearn library with the text reviews that have punctuations stripped off yields an accuracy of 0.3993.

5.2 Naïve Bayes

a. With tf-idf: *Accuracy: 0.5237*

b. With Bag of Words: *Accuracy: 0.5171*

5.3 Logistic Regression

a. With tf-idf

Regularizer	0.01	0.1	1	10	100
Accuracy	0.6564	0.6562	0.6562	0.**6563**	0.6563

The highest accuracy occur with C = 10.

b. With Bag of Words

Regularizer	0.01	0.1	1	10	100
Accuracy	0.6553	0.6553	0.6554	0.**6554**	0.6554

The highest accuracy occur with C = 10.

5.3 Linear SVC

a. With tf-idf

Regularizer	0.01	0.1	1	10	100
Accuracy	0.6395	0.**6440**	0.6274	0.5927	0.5667

The highest accuracy occur with C = 0.1.

b. With Bag of Words

Regularizer	0.01	0.1	1	10	100
Accuracy	0.6426	0.6414	0.**6446**	0.6216	0.5648

The highest accuracy occurs with C = 1.

5.4 KNN with tf-idf

For the hyper-parameter tuning, we first try to to find an optimal k, in the range of [1,20], with 3-fold cross validation, to minimize the misclassification error. The optimal number of neighbors is 19 where misclassification error for this k is 0.355.

k value	16	19
Accuracy	0.5804	0.**5809**

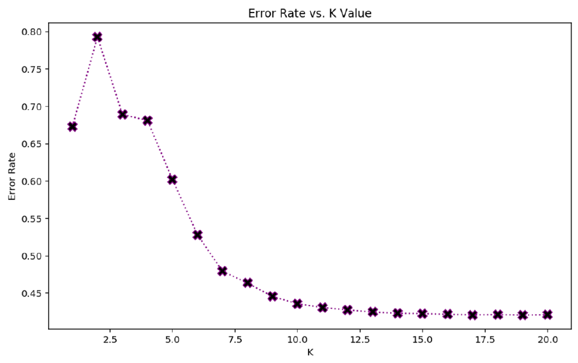
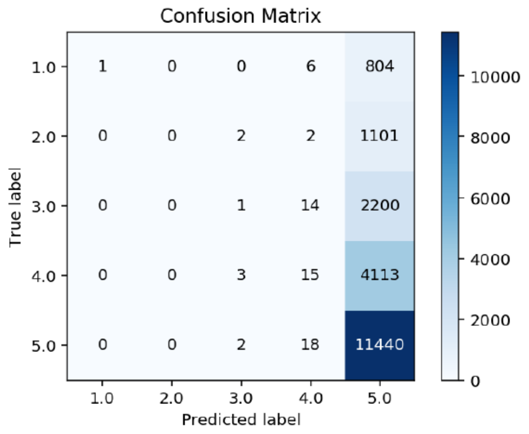


Figure 10: error rate for K values in range 1 to 20



	1.0	2.0	3.0	4.0	5.0
1.0	1	0	0	6	804
2.0	0	0	2	2	1101
3.0	0	0	1	14	2200
4.0	0	0	3	15	4113
5.0	0	0	2	18	11440

Figure 11: confusion matrix for KNN

Observed from the confusion matrix, most correctly predicted labels are 5.0. By calculating the (overall true predictions using diagonal line) / (overall review numbers), we got 0.5809.

5.5 Latent-factor Model

The gradient descent optimization for latent factor model converged pretty fast. The hyper-parameter values we used were λ=5.25, κ=0.1

It converged in nearly 59 iterations, with a good MSE error rate.

The model provided good results for λ ranging from 5.25 to 6, of which 5.25 gives the lowest mean square error 1.079.

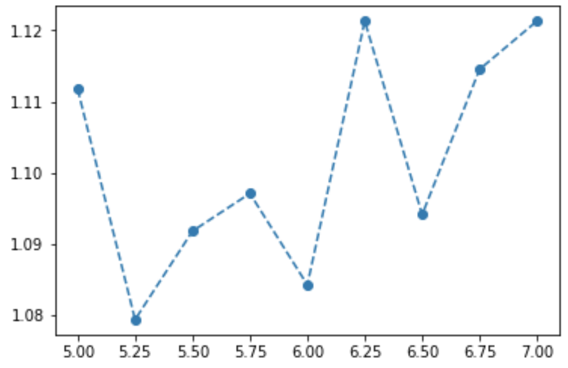


Figure 12: Latent Factor parameter tuning

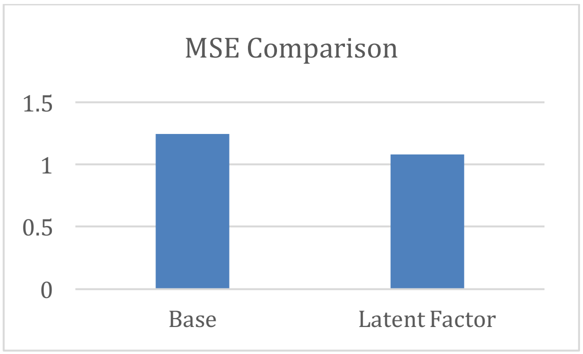


Figure 13: Latent Factor model MSE

5.6 Cross-model Comparison

Model	**Accuracy**
Base	0.3993
Bayes (BoW)	0.5171
Bayes (tf-idf)	0.5237
LinearSVC (BoW)	0.6446
LinearSVC (tf-idf)	0.6440
Logistic (BoW)	**0.6554**
Logistic (tf-idf)	**0.6563**
KNN	0.5809

Table 2: performance of different models

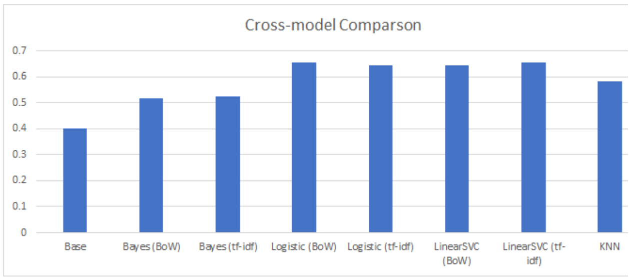


Figure 14: performance of different models

5.7 Conclusion

Given the comparison, we can see that logistic regression with tf-idf has the best performance in the 5-category review rating prediction task, rivaling LinearSVC. KNN gives medium accuracy, and Bayes Model has relatively poor performance (almost 0.5). Since latent factor only draws information from userID and reviewID, its performance is somewhat limited, with only a small advantage in terms of MSE over the baseline model.

However, we should also observe that the difference in accuracy between linear SVC and logistic regression is not very significant – the difference between the best accuracy of the two model with the same feature vectors is only around 1%. There reason accounting for this problem could be that the underlying implementation of the two type of classifier in sklearn uses the same liblinear library, and therefore the underlying classifier used by logistic regression could be similar to linear SVC. Therefore, the performance of the two models are similar.

We can also observe that the tf-idf representation had a better performance when used with Naïve Bayes and linear SVC, while the simple Bag-of-Word representation works slightly better with logistic regression. However, the difference in the accuracies produced by the same model on the two representation is also small. The reason could be that words with a high tf-idf could also have a high frequency in the same document; words with a low tf-idf could have a low frequency in the same document. Therefore, the outlook of the two representation could be somewhat similar in terms of this dataset, and therefore yields similar results.

6) REFERENCE

[1] R. He, J. McAuley, Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering
[2] Thorsten Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features
[3] Xiang Zhang, Yann LeCun, Text Understanding from Scratch
[4] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, Bag of Tricks for Efficient Text Classification
[5] Lalit Sachan, Logistic Regression vs Decision Trees vs SVM: Part II
[6] Adi Bronshtein, A Quick Introduction to K-Nearest Neighbors Algorithm