Training a Model to Understand Bias In Yelp Review Utility

Adele Chui
Department of Systems Design Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1
Email: aschui@edu.uwaterloo.ca

Arumoy Shome
Department of Systems Design Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1
Email: ashome@edu.uwaterloo.ca

***Abstract*—Yelp reviews can be manually labelled by users as useful as a way to filter what is informative and what is not for others. However, is there bias in this labelling process? Research indicates that humans can be easily swayed by other indicators outside of the utility of a piece of information in other scenarios. A Naive Bayes classifier, a Linear SVM classifier, and a random forest classifier were used to construct four separate models that attempt to predict whether a review was useful. It was found that the model that used review text, previous useful vote count, and cool vote count provided the most accurate results, highlighting that users are biased by things outside of review text when labelling a review as useful. Additionally, while literature uses random forest classifiers to achieve an accuracy of 95%, an accuracy of 99% was achieved here with a Linear SVM classifier.**

Keywords: machine learning, Yelp, reviews, automatic classification, utility.

1. Problem & Motivation

As a publisher of crowd-sourced reviews about local businesses, Yelp [1] exists in a space where the value of its service depends directly on the value of the reviews provided by users. In Yelps system, a user can manually upvote reviews as useful to indicate to other users that this piece of text provided meaningful information about a business from their perspective. However, relying on users to manually indicate utility can introduce bias that is unrelated to the actual relevance of the review text. For local businesses and users who are increasingly reliant on such systems over traditional food critiques, the legitimacy of a review can make or break an experience.

The idea of usefulness is a complicated one, affected by a users perception and needs. The Mirriam-Webster dictionary defines this idea as a level of fitness for some purpose or worth to some end [2]. In the case of online reviews, the utility of the text to a user is how well it provides relevant information about the interaction with the business in question. However, because different users have different needs, the value of the supposed relevant information within a review will differ. Additionally, there is a known difference in instant, real-time, utility and retrospective evaluation, remembered, utility [3]. For a system like Yelp, while user labels of value can provide an additional layer of filtering, it is clear that differences exist in exactly how usefulness can be defined between individuals.

Knowing that there is a distinct difference between user ideas of usefulness, bias compounds the issue. Herd instinct and social influence are two examples of common biases that are unrelated to the actual information within a review but lead to biased results. Herd instinct was first presented by Trotter to define the innate psychological dependence of humans on their social group [4]. Knowing that people are motivated to achieve goals in the most effective and rewarding manner possible and that people are social creatures, individual thoughts are often coloured by the opinions of the group [5]. Such influences are so strong that a study done where comments on a news-aggregate site were randomly manipulated with a small positive or negative judgement caused positive comments to be skewed upwards by 25% [6]. Not only did it change the mean of the ratings distribution, those that were manipulated were 30% more likely than control comments to reach the highest rating level. Clearly, existing research indicates that bias is a possible problem with online reviews. With no consideration for the review text itself, a study using over 37 000 Amazon UK reviews found that review readability made the largest difference in the user-rated helpfulness of a review [7]. With bias present even in considering the usefulness of a review, it is clear that the current system fails users in providing them with the legitimately valuable information they desire.

In this paper, a model is created that predicts a reviews usefulness based on review-text only as a control. Other models that incorporate other non-review-text related features are then created and compared against the control in order to identify if bias exists when users manually label reviews as useful. If bias does exist, knowing the specific features that cause this can be used to better tweak Yelps ecosystem to surface unbiased information for users. Do users read a review before voting it useful? Or is their decision based on other factors that are unrelated to the information within the review itself such as the cool, useful, or funny votes previously received?

2. Background

There exists a large amount of existing research into online reviews based on the Yelp dataset. The dataset itself is open source and updated with new data annually, encouraging new developments in the review analysis space each year [8].

A literature review found that while there are a variety of different techniques [9] [10] being used to analyze review data, random forest classifiers are incredibly popular [11] [12]. One group used Bag-of-Words on review data combined with features from user and business datasets to predict the required numeric value of the usefulness of a review with a batch mode localized weighted regression model. This localized regression approach resulted into RMSLE of 0.47769 [9]. Others compared SVM with logistic regression with Lasso to identify that the latter was the best with lower training error [10]. However, in comparison, using a random forest model with 150 estimators, the square root of original feature numbers to get maximum features, with ten minimum number of leaves to split, and no maximum depth lead to a model accuracy of 0.698926 [11]. Further improvements using random forest prediction with five fold cross validation were able to achieve an accuracy rate of 79% [12]. Another group took this even further and was able to achieve 95% accuracy in correctly labelling a review as useful using this random forest classifier approach [13].

In parallel with this research into predicting the usefulness of a review is developing an understanding of what skews bias in online reviews.

Studies use Amazon and Yelp data to look at what features specific to these two ecosystems cause bias in user perception of review value. Typically, a review is seen relative to other reviews. In a study using over 4 million Amazon book reviews for roughly 675 000 books, the perceived usefulness of reviews was dependent on its relationship to other reviews, not only its content [14]. Information about the reviewer themselves can also affect the perceived utility of a review [15]. Yelp restaurant reviews had their usefulness affected by user-based features more than detailed information within the review about the features of the restaurant [16]. More specifically, another group looking at over 72 000 Yelp reviews found that neutral and negative reviews by locals were seen as more useful, even if the information within the review was not necessarily valuable [17].

The novelty of this paper is found in the granular approach to which Yelp dataset-specific features are considered. Instead of a broad generalization of what creates bias, the focus is on features with high correlation with review usefulness in order to understand if they bias users. Do users actually read review before voting it useful? Or is their decision based on the votes the reviews had already received?

TABLE 1. SUMMARY OF DATA FEATURES

	funny	stars	useful	cool
count	5.261669e+06	5.261669e+06	5.261669e+06	5.261669e+06
mean	5.091960e-01	3.727740e+00	1.385085e+00	5.860916e-01
std	2.686168e+00	1.433593e+00	4.528727e+00	2.233706e+00
min	0.000000e+00	1.000000e+00	-1.000000e+00	-1.000000e+00
25%	0.000000e+00	3.000000e+00	0.000000e+00	0.000000e+00
50%	0.000000e+00	4.000000e+00	0.000000e+00	0.000000e+00
75%	0.000000e+00	5.000000e+00	2.000000e+00	1.000000e+0
max	1.481000e+03	5.000000e+00	3.364000e+03	1.105000e+03

TABLE 2. CORRELATION BETWEEN DIFFERENT POSSIBLE FEATURES

	funny	stars	useful	cool
funny	1.000000	-0.048866	0.621663	0.661669
stars	-0.048866	1.000000	-0.077122	0.044828
useful	0.621663	-0.077122	1.000000	0.677069
cool	0.661669	0.044828	0.677069	1.000000

3. Methods

3.1. Datasets & Sources

The Yelp dataset is a subset of Yelps businesses, reviews, and user data updated annually for independent use. It is available in both JSON and SQL files and is split into six sections: photos, tips, business check-ins, user data, review data, and business data. The most relevant subset for the purposes of this report is the review data, which includes labels for useful, cool, stars, and funny. These labels are manually applied to reviews by users.

The dataset contains over 5 million reviews for over 174 000 businesses. For the purposes of this project, the data was loaded into a dataframe.

3.2. Feature Selection & Data Processing

Beyond just review text data, the useful count, funny count, and cool count could also be used in this approach. By checking correlation between a feature and the useful count, the top two features were found. This was the useful and cool counts, which has a 0.62 and 0.66 correlation with the utility of a review.

Reviews were labelled as useful if they received one or more useful votes and labeled not useful if they did not receive a vote. In order to ensure unbiased models, data was randomly sampled to ensure that all labels used had a relatively equal number of data points. This will ensure that models have a relatively equivalent number of examples from which to learn.In this case, 10 000 reviews were chosen, with 50% of them labelled as useful and 50% labeled as not useful. A random sample of reviews was used to reduce time and geography bias, in order to reduce the influence of local or temporary trends that could skew review content.

TABLE 3. SNIPPET OF DATA AFTER "USEFUL" & "NOT USEFUL" SORTING

text	useful	cool
I can't begin to explain what an amazing place...	0	0
I found Jen at Essentials on Yelp after one so...	0	0
Came in here for for a quick beer. Bartenders ...	0	0
Absolutely amazing selection and variety of pl...	2	2
Use to love this place , hadn't been here in a...	0	0

Because the review data itself is text based, it must be processed. In this approach, the text was parsed to remove words in a process known as tokenization. A custom analyzer method was created, `review_process`, to remove all punctuation and new line characters. The words were then encoded for the machine learning approach outlined below in a process called feature extraction. Initially, scikit-learns `CountVectorizer` was used to tokenize the reviews and build a vocabulary of known words [18]. However, this was later changed to scikit-learns `TfidfVectorizer`, which converts text into TF-IDF features in an equivalent process to using a `CountVectorizer` followed by a `TfidfTrasnformer` [19]. This removed accents, stopwords, and converted everything to lowercase.

All zero-variance features were removed in order to improve algorithm performance.

3.3. Machine Learning Approach

The current state-of-the-art approach uses a random forest classifier to achieve 95% accuracy in its models ability to predict the usefulness of a Yelp review [13]. Knowing that the random forest approach has been thoroughly explored, a few different machine learning algorithms were implemented and compared against an implementation of random forest. In this case, a multinomial naive bayes classifier and a linear SVM classifier were tested against a random forest classifier.

A Naive Bayes Classifier is a simple classifier that is quick to implement. This is a simple probabilistic classifier based on applying Bayes theorem to features. It provides a benchmark to compare against the state of the art random forest classifier.

Listing 1. Naive Bayes classifier implementation

```
MultinomialNB(alpha=1.0,
      class_prior=None,
      fit_prior=True)
```

A linear SVM classifier has been used before to predict the stars associated with Yelp reviews and was modified to predict the usefulness of a Yelp review [20]. Linear SVM classifiers have not been used extensively in the review usefulness space but have the potential to perform very well.

Listing 2. Linear SVM classifier implementation

```
LinearSVC(C=1.0, class_weight=None,
      dual=True, fit_intercept=True,
      intercept_scaling=1,
      loss='squared_hinge',
      max_iter=1000, multi_class='ovr',
      penalty='l2', random_state=None,
      tol=0.0001, verbose=0)
```

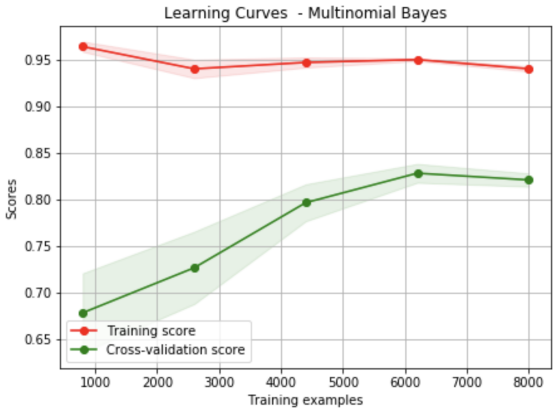

Figure 1. Naive Bayes learning curves

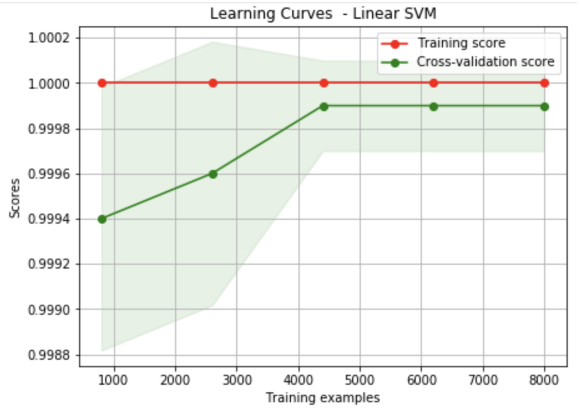

Figure 2. Linear SVM learning curves

Finally, a random forest classifier was used because it is the state-of-the-art in this space. Research has shown that random forest classifiers are capable of achieving 95% accuracy in predicting the usefulness of a Yelp review [13].

Listing 3. Random Forest classifier implementation

```
RandomForestClassifier(bootstrap=True,
      class_weight=None, criterion='gini',
      max_depth=None, max_features='auto',
      max_leaf_nodes=None,
      min_impurity_decrease=0.0,
      min_impurity_split=None,
      min_samples_leaf=1,
      min_samples_split=2,
      min_weight_fraction_leaf=0.0,
      n_estimators=10, n_jobs=1,
      oob_score=False, random_state=None,
      verbose=0, warm_start=False)
```

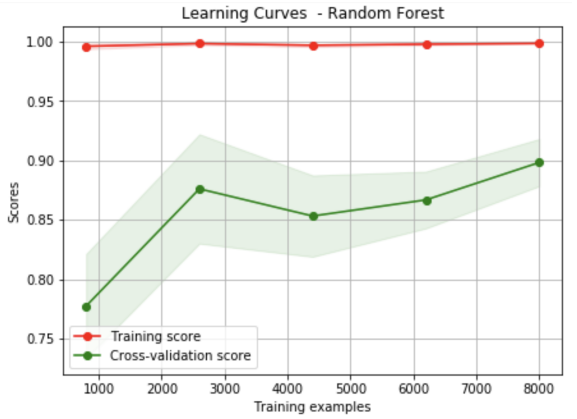

Figure 3. Random Forest learning curves

By comparing multiple models using different pieces of the dataset but with the 3 same machine learning approaches, a better understanding of which features lead to better performance and better match reality can be found.

K-fold validation was used to improve accuracy. The `sklearn.model_selection cross_val_score` was used with $k = 5$ folds [21]. This trains and validates the classifier with $k - 1$ folds and tests with the remaining one. This automates the process to split testing and training sets from the dataset.

3.4. Training

The first model was trained to predict if a review is useful based on only the text of the review. This used 10 000 data points and was built with all three machine learning approaches - Naive Bayes, Linear SVM, and Random Forest.

The second model was trained using both the review text and the useful votes. Recall that useful votes were highly correlated with reviews being voted more useful. Because this used multiple data types, data must be handled differently than with the first model. The useful column for both the training and data sets were converted into sparse matrices. These were normalized with the default L2 normalizer provided by `TfidVectorizer`. The useful sparse matrices are stacked with the text matrix horizontally in order to increase the number of features used in training. All three machine learning approaches were used again.

The third model was trained with review text and cool votes. Recall that cool votes had the highest correlation with a reviews utility as indicated by users. The same data transformation as used for the second model were also used here. All three machine learning approaches were used.

The fourth and final model used review text along with both useful and funny vote counts. Previously used data transformation techniques were applied again to train the three machine learning approaches.

TABLE 4. COMPARING ACCURACY ACROSS MODELS

Models	Multinomial Naive Bayes	Linear SVM	Random Forest
Model 1	57.45%	56.80%	57.79%
Model 2	58.07%	63.15%	84.49%
Model 3	57.66%	59.56%	65.06%
Model 4	82.10%	99.99%	86.41%

4. Results

The goal of this project was to determine whether useful votes received by reviews on Yelp are biased. To understand the presence of any such relationship, training was carried out in four stages:

1) To establish a benchmark, supervised learning of a model was performed on review text.
2) Next, the useful votes previously received by reviews are also added during training.
3) For the third model, the model was trained with review text and cool votes which has the highest correlation with useful votes.
4) Finally, the fourth model was trained with text, useful and cool votes.

The results of each of the four models is shown in Table 4.

5. Conclusion & Recommendations

It can be observed that reviews that obtained a previously high number of useful votes were more likely to be voted useful in the future. Model 1 and Model 3 share similar scores implying that cool votes do not cause bias towards usefulness of a review. Finally, Model 4 had the highest scores meaning useful and cool votes together cause maximum bias. Thus, in conclusion while readers may read review text, it is clear that reviews that have already been voted cool and useful will continue to be voted useful.

While it is still hard to conclude if users are actually reading reviews and then using solely their judgement to decide if a review is useful, the results indicate that there is significant bias provided by useful votes and to a lesser extent, cool votes.

Additionally, while existing literature indicates that a random forest approach works best at predicting usefulness, the work done in this paper actually indicates this is not always the case. While the random forest classifier did provide the highest accuracy when looking at text only, or text plus useful count or text plus cool count, the Linear SVM classifier actually performed the best when both useful and cool votes were included along with the review text. Since users are not just looking at review text solely or just looking at one type of vote and the review text when considering if a review is useful, it can be argued that the Linear SVM classifier best models what information is actually used by users since it provides the highest accuracy in the fourth model.

However, there are several recommendations that can be made to improve the results of this experiment. First, feature selection can be helpful after feature extraction as the number of data points increases. For the purposes of this paper, 10 000 data points were used. However, as this number increases, the right feature selection could possibly reduce classifier training time, reduce overfitting by generalizing the models, and reducing model complexity. Label generation could also be improved by using k-means to identify the labels Presently a Naive methodology has been used to determine the labels for the useful votes. A better solution would be to use k-means to determine appropriate labels on the useful votes.

Acknowledgments

The authors would like to thank Professor H. Tizhoosh for his insights into machine intelligence. We acknowledge the support provided by the previous works referenced - without their work, this paper would not be possible.

References

[1] "Yelp", Yelp, 2018. [Online]. Available: https://www.yelp.com/. [Accessed: 08- Apr- 2018].

[2] "Definition of UTILITY", Mirriam-Webster, 2018. [Online]. Available: https://www.merriam-webster.com/dictionary/utility. [Accessed: 09- Apr- 2018].

[3] D. Kahneman, P. Wakker and R. Sarin, "Back to Bentham? Explorations of Experienced Utility", The Quarterly Journal of Economics, vol. 112, no. 2, pp. 375-406, 1997.

[4] R. Raafat, N. Chater and C. Frith, "Herding in humans", Trends in Cognitive Sciences, vol. 13, no. 10, pp. 420-428, 2009.

[5] R. Cialdini and N. Goldstein, "Social Influence: Compliance and Conformity", Annual Review of Psychology, vol. 55, no. 1, pp. 591-621, 2004.

[6] S. Aral, "The Problem With Online Ratings", MIT Sloan Management Review, vol. 55, no. 2, pp. 47-52, 2014.

[7] N. Korfiatis, E. Garca-Bariocanal and S. Snchez-Alonso, "Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content", Electronic Commerce Research and Applications, vol. 11, no. 3, pp. 205-217, 2012.

[8] "Yelp Dataset", Yelp, 2018. [Online]. Available: https://www.yelp.ca/dataset/. [Accessed: 09- Apr- 2018].

[9] R. Shen, J. Shen, Y. Li and H. Wang, "Predicting usefulness of Yelp reviews with localized linear regression models", 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016.

[10] X. Liu, M. Schoemaker and N. Zhang, "Predicting Usefulness of Yelp Reviews", Stanford University, Stanford, 2014. http://cs229.stanford.edu/proj2014/Xinyue%20Liu,%20Michel%20Schoemaker,%20Nan%20Zhang,Predicting%20Usefulness%20of\%20Yelp%20Reviews.pdf

[11] H. Zhang, X. Li and K. Ying, "Reviews Usefulness Prediction for Yelp Dataset", UC San Diego, San Diego, 2014. https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a040.pdf

[12] M. Barakat, "Predicting the Usefulness of a Yelp Review Using Machine Learning", 2018. [Online]. Available: https://rstudio-pubs-static.s3.amazonaws.com/133133_fd7291a46ff547288a25af57b6b35100.html. [Accessed: 07- Apr- 2018].

[13] A. Ghenai, "What makes a review useful, funny or cool on Yelp.com", Project report for CS886: Applied Machine Learning, 2015.

[14] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg and L. Lee, "How opinions are received by online communities", Proceedings of the 18th international conference on World wide web - WWW '09, 2009.

[15] P. Racherla and W. Friske, "Perceived usefulness of online consumer reviews: An exploratory investigation across three services categories", Electronic Commerce Research and Applications, vol. 11, no. 6, pp. 548-559, 2012.

[16] L. Li, K. Zhang, Q. Zhou and C. Zhang, "Toward Understanding Review Usefulness: A Case Study on Yelp Restaurants", iConference 2016 Proceedings. 2016.

[17] J. Neumann, D. Gutt, D. Kundisch, and D. van Straaten, When Local Praise Becomes Cheap Talk - Analyzing the Relationship between Reviewer Location and Usefulness of Online Reviews. Proceedings of the Multikonferenz Wirtschaftsinformatik 2018 (MKWI), Lneburg, Germany, 2018

[18] "sklearn.feature_extraction.text.CountVectorizer", scikit-learn 0.19.1 documentation, 2018. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Accessed: 13- Apr- 2018].

[19] "sklearn.feature_extraction.text.TfidfVectorizer", scikit-learn 0.19.1 documentation, 2018. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. [Accessed: 13- Apr- 2018].

[20] G. Dwyer, "Predicting Yelp Stars from Reviews with scikit-learn and Python — DevelopIntelligence Blog", Develop Intelligence, 2018. [Online]. Available: http://www.developintelligence.com/blog/2017/03/predicting-yelp-star-ratings-review-text-python/. [Accessed: 09- Apr- 2018].

[21] "sklearn.model_selection.cross_val_score", scikit-learn 0.19.1 documentation, 2018. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html. [Accessed: 13- Apr- 2018].

Appendix A.
Code Listing

Listing 4. Random sampling of data

```
THRESHOLD = 1
SAMPLE_SIZE = 5000

reviews = pd.read_csv('dataset/review.csv', usecols=['text', 'useful', 'cool'])

useful_reviews = reviews.loc[reviews['useful'] >= THRESHOLD].sample(SAMPLE_SIZE)
not_useful_reviews = reviews.loc[reviews['useful'] < THRESHOLD].sample(SAMPLE_SIZE)
print("useful_reviews shape: ", useful_reviews.shape)
print("not_useful_reviews shape: ", not_useful_reviews.shape)

reviews = shuffle(pd.concat([useful_reviews, not_useful_reviews]))
```

Listing 5. Preprocessing of data

```
#!/usr/bin/env python

# # Preprocessing
#
# For `text` a custom `analyzer` method namely `review_process` is written which:
# 1. removes all punctuations and
# 3. removes new line characters (escape sequence)
#
# The feature extraction (`TfidfVectorizer`) class is then set to:
# 2. remove accents
# 2. remove all stopwords
# 3. lowercase all words
#
# For `useful` votes:
# 2. remove df row if `useful` is `NaN`

reviews.dropna(inplace=True)
print("reviews shape after dropping NaN values: ", reviews.shape)

import string
import re

RE_NEWLINE = '\n+'
PUNCTUATIONS = string.punctuation

def review_process(review):
    no_newline = re.sub(RE_NEWLINE, '', review)
    no_punc = ''.join([char for char in no_newline if char not in PUNCTUATIONS])

    return no_punc

vect = TfidfVectorizer(strip_accents='ascii', preprocessor=review_process,
stop_words='english')
```

Listing 6. Generating labels for data

```
import math

NOT_USEFUL = 0
USEFUL = 1
```

```
def labeler(vote):
    if math.floor(vote) < THRESHOLD:
        return NOT_USEFUL
    else:
        return USEFUL

reviews['label'] = reviews['useful'].apply(labeler)
```

Listing 7. Training Validation Testing and Scoring of uniform dtypes

```
X1 = vect.fit_transform(X['text'])
print("shape of X1 after feature extraction: ", X1.shape)
print("type of X1: ", type(X1))

mn_scores = cross_val_score(mn_clas, X1, Y, cv=5)
svc_scores = cross_val_score(svc_clas, X1, Y, cv=5)
rf_scores = cross_val_score(rf_clas, X1, Y, cv=5)

print("Naive Bayes score: ", print_percent(mn_valid_scores.mean()))
print("Linear SVC score: ", print_percent(svc_valid_scores.mean()))
print("Random Forest score: ", print_percent(rf_valid_scores.mean()))
```

Listing 8. Training Validation Testing and Scoring of mixed dtypes

```
#!/usr/bin/env python


# # Model 2: training with text and useful votes
#
# Since X now contains mixed dtypes, we need to be a bit clever. First, we
# convert 'useful' column for train and test into sparse matrix and *I2*
# normalize them (I2 is the default for 'TfidfVectorizer' so that's what we use
# here as well).

# The rest is straight forward, we create our updated X by stacking the 'useful'
# sparse matrices with X horizontally (such that the number of feature
# increases).

useful_sparse_train = normalize(csr_matrix(X['useful']))
print("type of useful_sparse_train: ", type(useful_sparse_train))
print("shape of useful_sparse_train: ", useful_sparse_train.shape)

X2 = hstack([X1, useful_sparse_train.T])
print("shape of X2: ", X2.shape)

mn_scores = cross_val_score(mn_clas, X2, Y, cv=5)
svc_scores = cross_val_score(svc_clas, X2, Y, cv=5)
rf_scores = cross_val_score(rf_clas, X2, Y, cv=5)

print("Naive Bayes score: ", print_percent(mn_scores.mean()))
print("Linear SVC score: ", print_percent(svc_scores.mean()))
print("Random Forest score: ", print_percent(rf_scores.mean()))
```