Data Imbalance Processing Based on Amazon Food Reviews

BU.520.710.F4. Big Data Machine Learning

Spring 2021

Team Members:

Ian Kamar, Youyang Zhou, Yunhao Dou, Yi Ding, Hanqiu Xu

1. Introduction

Many algorithms have a basic assumption that the data is uniformly distributed. When we apply these algorithms directly to real data, we cannot get the ideal result in most cases. The actual data is often very uneven distribution. The problem of data imbalance, while not the most difficult, is definitely one of the most important. Therefore, this paper studies the treatment of data imbalance problem.

The paper is divided into four parts. And the first part is introduce which explain the goal of the research problem; The second part is data which explains the source of data. The third part is methods which divided into four parts, natural language processing methods, imbalanced data processing methods, random forest method and the result analysis comparison. The fourth part is the conclusion part.

2. Data

Data Source is https://www.kaggle.com/snap/amazon-fine-food-reviews The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon. The data is as follows,

Number of reviews: 568,454

Number of users: 256,059

Number of products: 74,258

Timespan: Oct 1999 - Oct 2012

Number of Attributes/Columns in data: 10

Attribute Information:

Id

ProductId - unique identifier for the product

UserId - unqiue identifier for the user

ProfileName

HelpfulnessNumerator - number of users who found the review helpful

HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not

Score - rating between 1 and 5

Time - timestamp for the review

Summary - brief summary of the review

Text - text of the review

And 'Text' is the feature variable, 'sore' is the label variable. In this paper, 2000 sample are extracted from the comments dataset and analyzed. Label variables of 2 and 1 into a Category, recorded as 0; 4 and 5 into a category, recorded as 0. The distributions of the label variables for (raw data and subset data) are shown in the following figure.

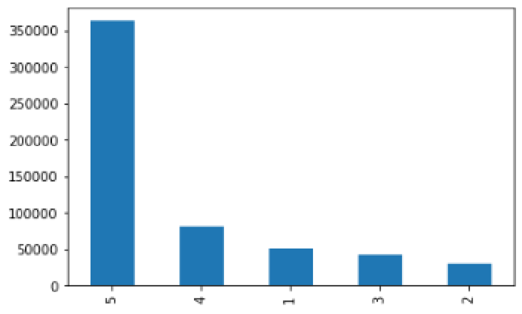


Figure 1: raw data label distribution

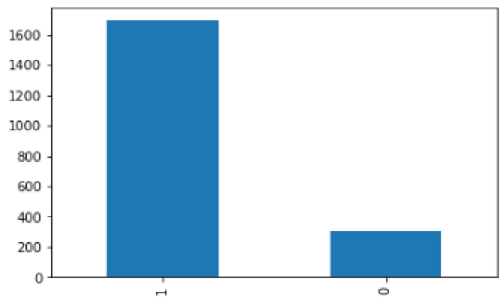


Figure 2: the paper data label distribution

It is not difficult to see from the chart that the data are seriously unbalanced, so it is necessary to study and analyze the problem of imbalance.

3. Methods

3.1 Word vector

Word Vector processing includes removing text cleaning and word vector. Here is a brief introduction to text cleaning. Text cleaning includes the following sections.

- Step 1: removing duplicates.
- Step2: Removing the designation of useless symbols, including html tags, any punctuation or limited special character sets, such as # , meaningless function words: "The" , "a" , "an" , "that" ' we' , 'they' , etc. .
- Step3: Look at the word composition. Check to see if the word is made up of English letters, not Alphanumeric, and see if the word is longer than 2(because the study found no adjectives in the 2 letters) .
- Step4: Convert words to lowercase letters.

The word vector process is the transformation of words into vectors. The bag of words model is used in the paper. The model in which words such as a sentence or document can be represented as words in a bag, regardless of grammar or word order. The vector representation of a document can directly sum the word vector representation of each word.

For example:

John likes to watch movies. Mary likes too

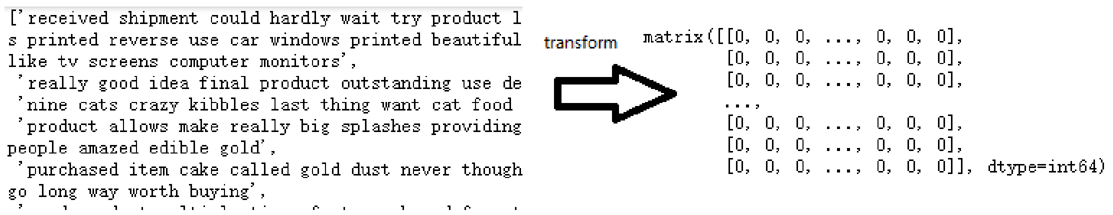
John also likes to watch football games.

The above two sentences can form a dictionary D, d={"John": 1, "likes": 2, "to": 3, "watch": 4, "movies": 5, "also": 6, "football": 7, "games": 8, "Mary": 9, "too": 10},

The vector representation of the first sentence is: [1,2,1,1,1,0,0,0,1,1]. And 2 represented likes, and so on.

The paper is implemented using the CountVectorizer function of the word bag model BOW in python. CountVectorizer uses the fit_transform function to convert words in text into a word-frequency Matrix. The Matrix element a[i][j] denotes the frequency of the j word in the i text. That is, the number of occurrences of each word. Using get_feature_names() function to see all text keywords, and toarray () function to see the results of the Word Frequency Matrix.

After the word vector process, the text becomes vector format, as shown below.



```
['received shipment could hardly wait try product l
s printed reverse use car windows printed beautiful
like tv screens computer monitors',
 'really good idea final product outstanding use de
 'nine cats crazy kibbles last thing want cat food
 'product allows make really big splashes providing
people amazed edible gold',
 'purchased item cake called gold dust never though
go long way worth buying',
```

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Figure 3: text—>word vector

3.2 Imbalanced data processing

The up-sampling (over-sampling) and down-sampling(negative-sampling) strategies are one of the basic methods to solve class imbalance problem. The upper sampling is to increase the number of a few samples, and the lower sampling is to decrease the number of samples in order to obtain a relatively balanced data set. The simplest up-sampling method can directly copy a few samples and add them to the sample set, and the simplest down-sampling method can directly take only a certain percentage of most samples as the training set.

The sampling strategy in this paper includes RandomOverSampler, SMOTE and SMOTETomek.

The data were divided into training samples and test samples according to the ratio of 8:2. The number of training samples was 1598. After three kinds of sampling, the number of training samples was 2720, and the number of positive and negative samples were 1360.

3.3 Random forest methods

Random Forest is an extension of Bagging. On the basis of constructing Bagging ensemble based on decision tree, random attribute selection is introduced into the training process of decision tree, that is, the traditional decision tree selects the best attribute in the attribute set of the current node (assuming there are d nodes) based on the information purity criterion, while in the random forest, for every node of the base decision tree, firstly, a subset containing k attributes is selected randomly from the attribute set of the node, and then the subset is selected according to the information criterion, then the base decision tree is constructed the same as the traditional decision tree, if k = 1, then each selection of attributes is purely random and independent of the information criterion, generally, $k = \log_2 d$ is recommended.

Each tree in a random forest is generated as follows:

1. If the training set size is n, randomly and randomly put back n training samples from the training set (this sampling method is called bootstrap method) as the training set of the tree; from this we can know that the training set of each tree is different and contains repeated training samples.
2. If the feature dimension of each sample is M, assign a constant m<<M, randomly select M feature from m features subsets, and select the best one from these m features each time splitting the trees.
3. Each tree grows to its fullest extent and is not pruned.

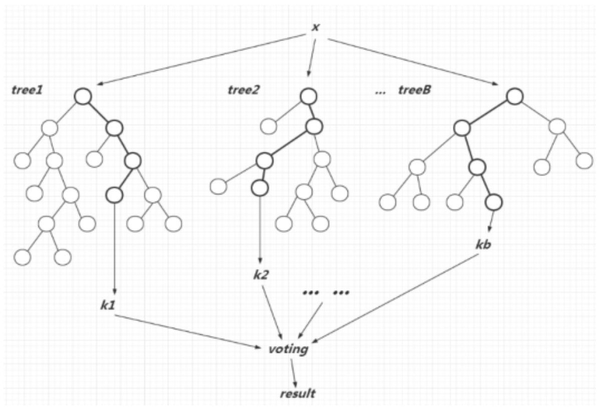
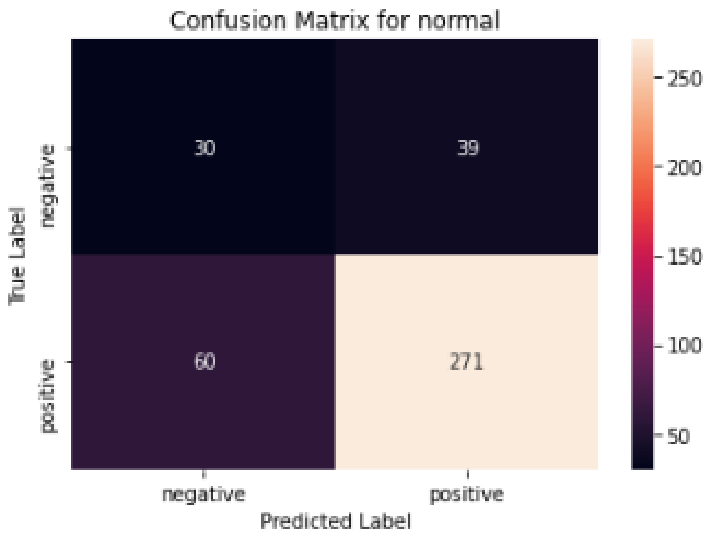


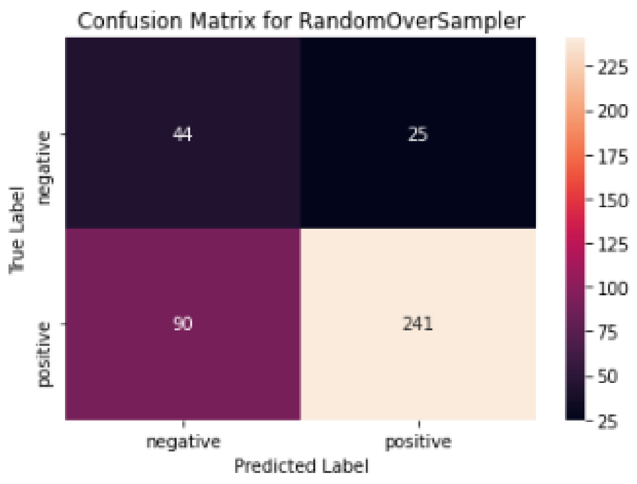
Figure 4: Random Forest

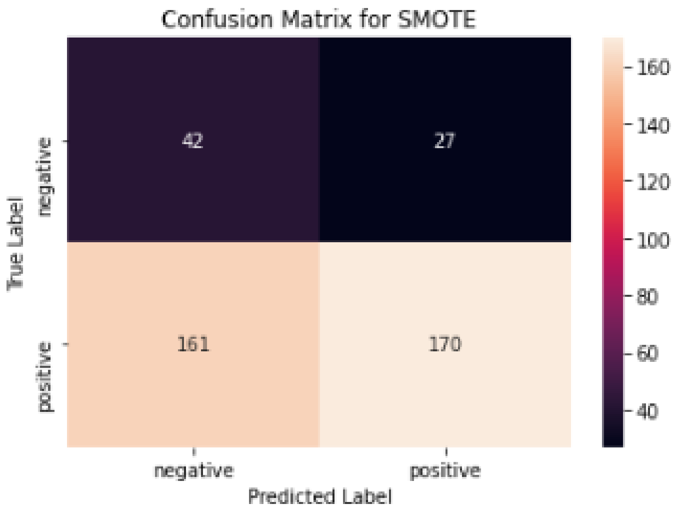
The raw training data and the sampling data of three sampling methods are used for random forest training.

3.4 Comparison

The comparison method used in this paper is the confusion Matrix, the ROC curve, four methods after training, respectively to predict, and then compare the results. The confusion Matrix and ROC curve are shown below.







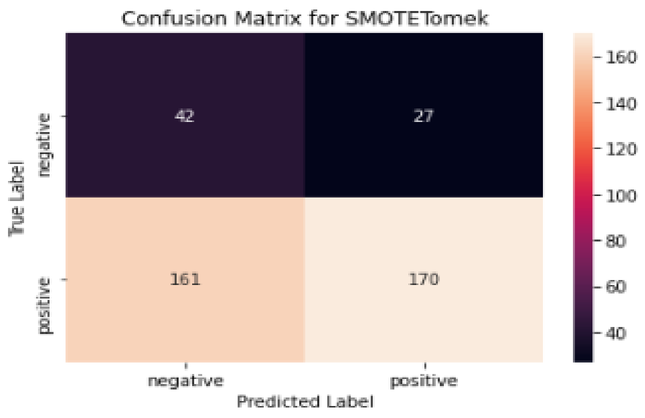


Figure 5: Confusion Matrix

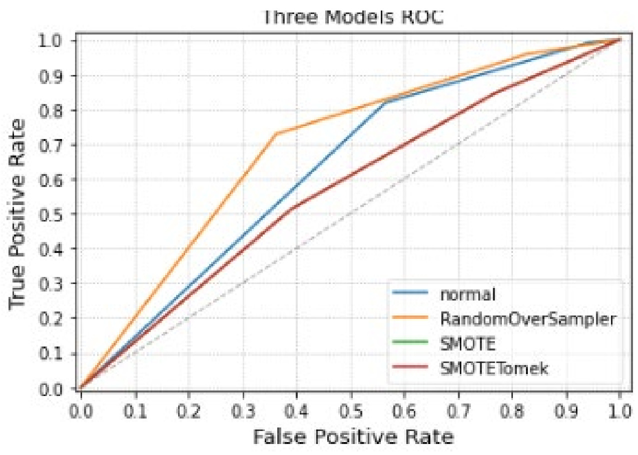


Figure 6: the ROC curve

The precision rates of the four methods were 0.874, 0.906, and 0.863 respectively. It is not difficult to see from the graph and the accuracy results that the random sampling method is better than the original data results. The other two sampling methods are no better than the original data method.

4. Conclusion

The two sampling methods are not good because of the possible sampling ratio, which is the problem to be considered in the future. Whether under-sampling or over-sampling is used, only the appropriate is the most important.

Reference

Data Source:

https://www.kaggle.com/snap/amazon-fine-food-reviews

https://github.com/krpiyush5/Amazon-Fine-Food-Review

Appendix

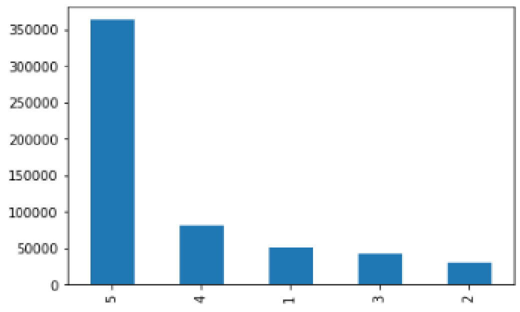


Figure 1: raw data label distribution

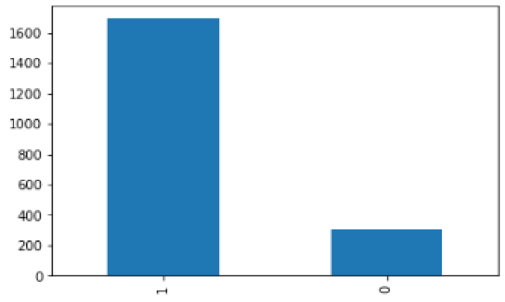
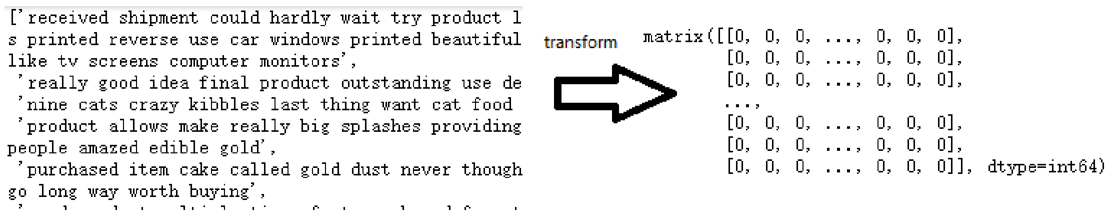


Figure 2: the paper data label distribution



```
['received shipment could hardly wait try product l
s printed reverse use car windows printed beautiful
like tv screens computer monitors',
 'really good idea final product outstanding use de
 'nine cats crazy kibbles last thing want cat food
 'product allows make really big splashes providing
people amazed edible gold',
 'purchased item cake called gold dust never though
go long way worth buying',
```

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Figure 3: text—>word vector

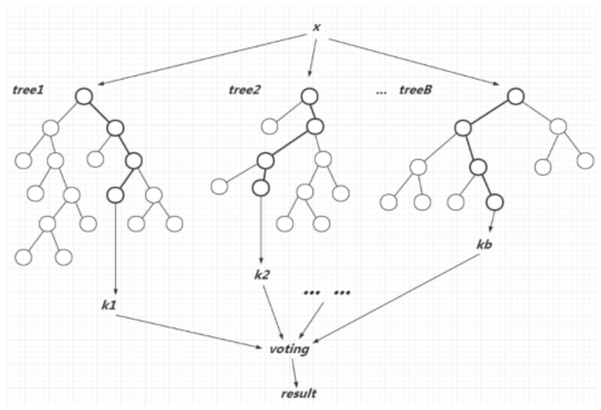
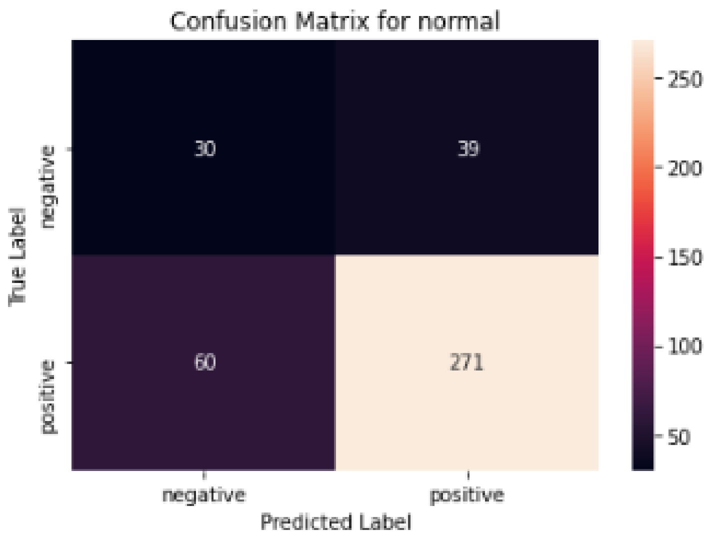
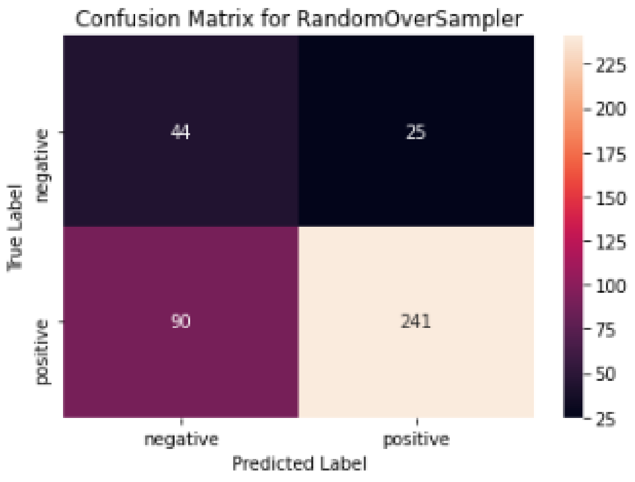
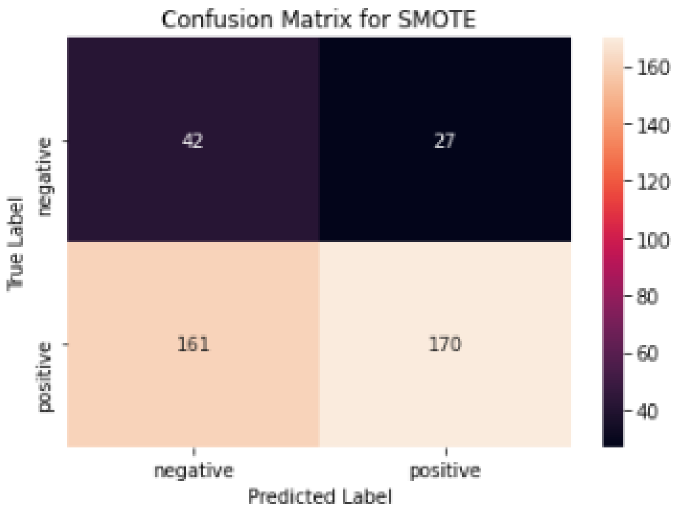


Figure 4: Random Forest


Confusion Matrix for normal
True Label
negative
positive
30
39
60
271
negative
positive
Predicted Label
250
200
150
100
50


Confusion Matrix for RandomOverSampler
True Label
negative
positive
44
25
90
241
negative
positive
Predicted Label
225
200
175
150
125
100
75
50
25


Confusion Matrix for SMOTE
True Label
negative
positive
42
27
161
170
negative
positive
Predicted Label
160
140
120
100
80
60
40

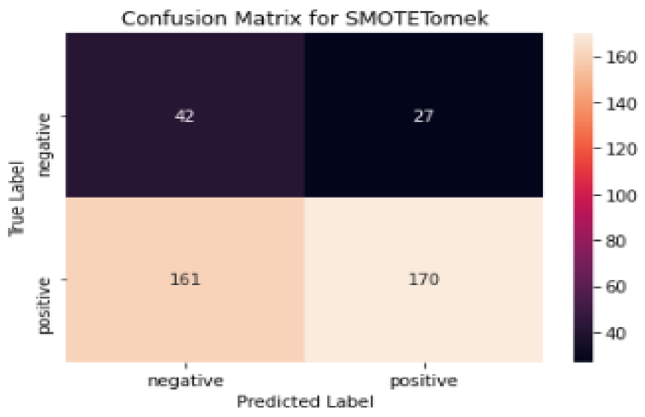


Figure 5: Confusion Matrix

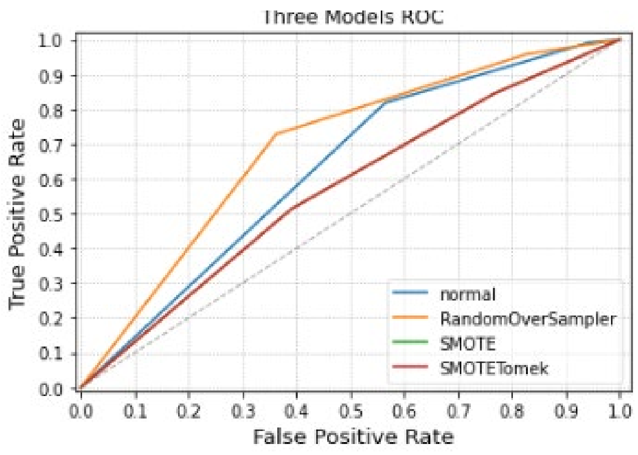


Figure 6: the ROC curve