HW6

Bernoulli and Multinomial Naïve Bayes in Sci-kit Learns

Bing-Je Wu

IST 736 – Text Mining

Table of Contents

[Introduction 2](#_Toc39867598)

[Analysis and Models 2](#_Toc39867599)

[About the Data 3](#_Toc39867600)

[Data Processing 4](#_Toc39867601)

[Models 7](#_Toc39867602)

[Sentiment 8](#_Toc39867603)

[Authenticity 12](#_Toc39867604)

[Results 16](#_Toc39867605)

[Conclusion 17](#_Toc39867606)

[Reference 17](#_Toc39867607)

# Introduction

Reviews are everywhere on the internet. People tend to write their thought for feeling about something. It could a though of a dish from a restaurant. Or, it could be a feeling about a movie. Even, it can be a complement for a seller from an e-commerce platform. There are tons of reviews produced every day, every second. Among the enormous reviews, it may have fake reviews with intention of destroying someone’s reputation. And, that is the cue for information science comes into play. Having the ability to classify the reviews good or bad, fake or true, becomes a good topic for researcher doing research.

Analysis and Models  
Two algorithms, ’MultinomialNB’ (Multinomial Naive Bayes) algorithm and ‘BernoulliNB’ (Bernoulli Naive Bayes)‘ from scikit learn package, were used to build models. Two transformers, CountVectorizer and TfidfVectorizer, were utilized for model building. Holdout test technique (train\_test\_split) and k-fold cross-validation approach (cross\_val\_score) were implemented for model selection. A pipeline (make\_pipeline) was built to get a smooth workflow. A random grid search cross-validation (RandomizedSearchCV) was applied to tune the parameters for getting the best score of the selected model.

An advanced feature engineering technique, combining the document term matrix and manual features, was also included for having more options on model selection. The technique utilized the ‘transformer’ class (FunctionTransformer) that is provided by sklearn library. The concept of the technique is to create a union (make\_union) with parallel transformers, a sequence of a pipeline (vectorizing text after a text feature transformer) and manual feature transformer. The union can be combined with an algorithm to build a pipeline in high level for model training and model evaluation. In this case, the recurrent preprocessing steps for building a model are vastly reduced. Figure 1 shows the example of Pipeline and FeatureUnion. Figure 2 exhibits the workflow of the data transformation in text with manual features.

**Figure 1**. Transformer Example

A picture containing photo, sitting, black, dark

Description automatically generated

**Figure 2.** Transformation Workflow

A picture containing dark

Description automatically generated

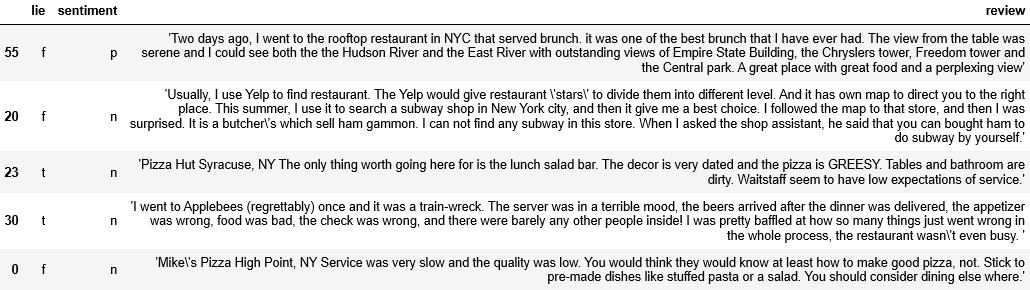
## About the Data

Data were assigned from online learning management system. It is a restaurant review dataset. The dataset contains two labels. One is ‘lie’ label; the other is ‘sentiment’ label. For the ‘lie’ label, it has two classes, ‘t or ‘f’. If a restaurant review is a lie, it will be labeled as ‘f’ (fake). On the other hand, if a review is not a lie, then, it will be labeled as ‘t’ (true). For the ‘sentiment’ label, it contains two classes, ‘n’ or ‘p’. If a review is negative, it will be assigned a label as ‘n’. Otherwise, it will be assigned a label as ‘p’ if it is a positive review.

**Table 1.** *Dataset*

|  |  |  |
| --- | --- | --- |
| Attribute | Description | Data Type |
| lie | Label of authenticity (true or fake, lie detection) | String |
| sentiment | Label of sentiment (negative or positive) | String |
| review | The text of restaurant review | String |

**Figure 1**. *Sample of data before cleaning*

**

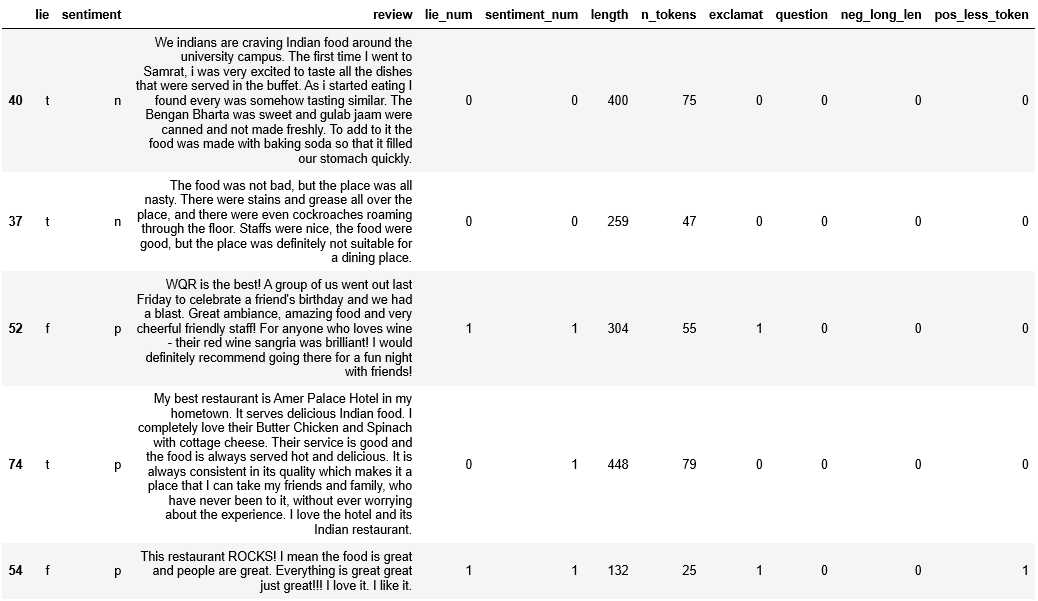
Data Processing  
To perform a regular text classification task, several cleaning steps must be done before building a model. By exploring the dataset, it is easy to notice that the ‘review’ column is not standard. Some back slashes are in the text. There are leading and trailing apostrophe in the string as well. It will be a problem when a ‘CountVectorizer’ function is applied to the dataset. Therefore, a customized function, ‘clean\_str()’, was applied to the dataset to remove the trailing and leading apostrophe and redundant black slashes. Two dictionaries, ‘lie\_mapper’ and ‘sentiment\_mapper’, were used as mapper to convert the code to numeric for getting the feature ranking in the later process.

Feature engineering technique is also implemented based on the current existing columns. 6 extra features were added to help classification task along with document term matrix. A ‘length’ column was created by using ‘len()’ built-in function applied on the ‘review’ column. The ‘n\_tokens’ column was added with two functions, ‘len()’ and ‘split()’ on the ‘review ‘column. Both ‘exclamat’ and ‘question’ columns were created to see if an exclamation mark or a question mark existed in the review content. Last, in order to find out if a review is a fake review or a true review, the ‘sentiment’ column and ‘length’ column need to be combined to create new columns, ‘neg\_lon\_len’ and ‘pos\_less\_token’. It is the human nature that people tend to tell a lie with a bunch of words and vice versa.

**Table 2.** *User defined functions for data cleansing and pre-processing*

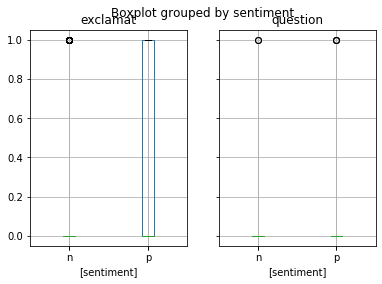
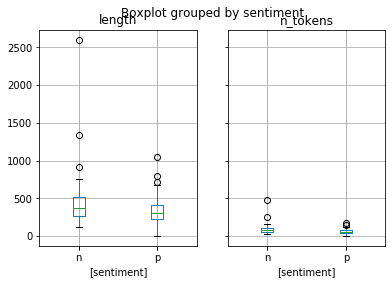
|  |  |  |
| --- | --- | --- |
| Function | Type | Explain |
| clean\_str() | Function | * Expect: A string with leading and trailing apostrophe and a back slash within the string. ex: " 'I found worm \in one of the dishes.' " * Modify: Remove the trailing and leading apostrophe and back slash * Return: Return a string |
| lie\_mapper | Dictionary | Convert ‘n’ (negative) to 0 and ‘p’ (positive) to 1 |
| sentiment\_mapper | dictionary | Convert ‘t’ (true) to 0 and ‘f’ (fake) to 1 |
| length | lambda | Count the length of the each of document (review) |
| n\_tokens | lambda | Count the number of words after applying ‘split()’ on each of document |
| exclamat | lambda | Return ‘1’ if an exclamation mark existed in the review, otherwise return ‘0’ |
| question | lambda | Return ‘1’ if a question mark existed in the review, otherwise return ‘0’ |
| neg\_long\_len | lambda | Return ‘1’ if a review is negative and has the length of the review over 400, otherwise return ‘0’ |
| pos\_less\_token | lambda | Return ‘1’ if a review is positive and has the ‘n\_tokens’ less than 50, otherwise return ‘0’ |

**Figure 2**. *Sample of cleaned data after cleaning*



After finishing the data cleaning process, an initial data exploratory analysis was conducted to find the useful features for classification task. Figure 3 shows the ’exclamat’ and ‘length’ columns have some distinction between negative review and positive review. It seems like it could be a good predictor in the classification tasks. Other features, such as ‘n\_tokens’ and ‘question’, have overlaps. Thus, it will not be added to the model building process. The ‘neg\_lon\_len’ and ‘pos\_less\_token’ were not be considered since it is derived from ‘sentiment’ column. Using those features will violate the data snooping bias in this case.

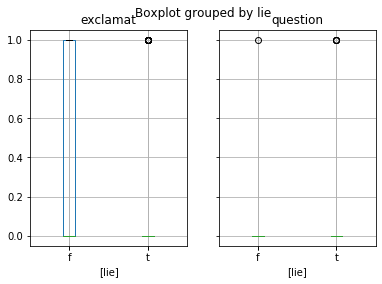
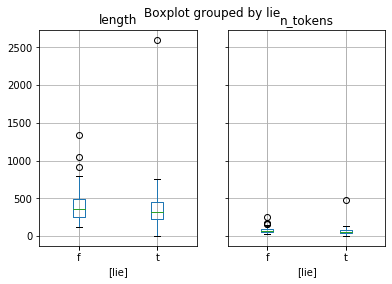
**Figure 3.** Boxplot for Sentiment

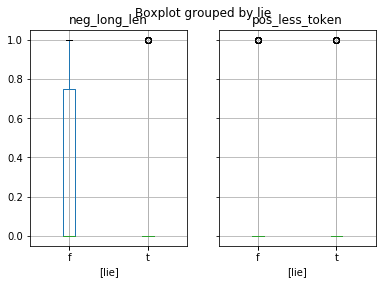


It is clear to see that ‘exclamat’ and ‘neg\_long\_len’ features have some distinction between ‘fake’ and ‘true’. Other columns, ‘length’, ‘n\_tokens’, ‘question’, ‘pos\_less\_token’ seem to be having overlaps. Thus, in the final step of model building, combining the document term matrix and manual features, the ‘exclamat’ and ‘neg\_long\_len’ will be added to build a better model.

Although the manual features were created, the traditional model building approach will be implemented first before adding any manual feature. Sometimes models without manual features could perform better than the models with manual features.

**Figure 4.** Boxplot for Lie





Models  
There are two prediction tasks. One is lie detection, predicting a review is true review or fake review. The other task is sentiment prediction, predicting a review is a positive or negative review. Naïve Bayes models were built for those tasks by implementing the ‘multinomialNB’ algorithm and ‘BernoulliNB’ algorithm from sklearn. The ‘multinomialNB‘ was implemented with two transformers, CountVectorizer and TfidfVectorizer. And the ‘BernoulliNB’ was implemented with CountVectorizer and CountVectorizer input setting of binary equal to true. A ‘’make\_prediction” user-defined function has been created to expedite the training process. After utilizing the ‘make\_prediction’ function to get the performance of each models, a best candidate model will be elected to re-trained with entire dataset having grid search technique for tuning the best parameter for the model. Then, an advanced approach will be applied with desired features along with document term matrix to build another series of models and go through the same training and selecting process by using ‘make\_prediction\_manual\_ft’ function to compare the metrics. Finally, another ‘grid\_result’ function will be applied to get the best combination of the parameters for the model with manual features.

**Table 3**. *User-defined functions for model building*

|  |  |
| --- | --- |
| Function | Explanation |
| make prediction | The function takes four input arguments, ‘X’, ‘y’, ‘vect’ and ‘clf’. The ‘X’ argument takes a pandas series that include text. ‘y’ is the label of the data. ‘vect’ is the object name of a vectorizer. ‘clf’ is the name of a classifier. The workflow of the function is as below:   1. Apply train\_test\_split for holdout test; X\_train, X\_test, y\_train, y\_test 2. Create a pipeline; pipe = vectorizer + classifier 3. Use Cross-Validation on pipe with X\_train 4. Validate with X\_test using ‘predict’ method on pipeline 5. Create confusion matrix and classification report |
| grid result | The function takes two input arguments, ‘pipe’ and ‘param\_grid’. The ‘pipe’ argument takes the pipeline object. The ‘param\_grid’ takes a dictionary that contain the parameter setting for model tuning. The function uses RandomizedSearchCV to find the best score and best combination of parameters. |
| get\_manual | The function takes a pandas dataframe as an input. It returns a pandas dataframe that is a subset of the input. The filter condition is based on the feature that is desired to be used in model training process. |
| get\_text | The function takes a pandas dataframe as an input. It returns a pandas series that contains text as the input. |
| make\_prediction\_manual\_ft | The function takes four input arguments, ‘X’, ‘y’, ‘vect’ and ‘clf’. The ‘X’ argument takes a pandas series that include text. ‘y’ is the label of the data. ‘vect’ is the object name of a vectorizer. ‘clf’ is the name of a classifier. The workflow of the function is as below:   1. Apply train\_test\_split for holdout test; X\_train, X\_test, y\_train, y\_test 2. Create a pipeline; pipe = [(get\_text\_ft + vectorizer) + get\_manual\_ft]+ clf 3. Use Cross-Validation on pipe with X\_train 4. Validate with X\_test using ‘predict’ method on pipeline 5. Create confusion matrix and classification report |

Sentiment   
Three combination of models were used throughout the training process. The first model was used BernoulliNB algorithm and CountVectorizer with the setting, ‘binary=true’ and ‘min\_df=5’. The second model was built on MultinomialNB with CountVectorizer having the setting, ‘min\_df=5’. The third model was utilized the same algorithm, MultinomialNB, to build the model with TfidfVectorizer having the setting, ‘min\_df=5’. The detail information was shown in Table 4 and Table 5. Among the three models, the third model (MultinomialNB with TfidfVectorizer) was performing the best with 91.3% on the test set. The random grid search result was included in the Table 5 showing that the re-trained model yielded the accuracy of 88.01% with entire dataset. The overall performance is satisfying.

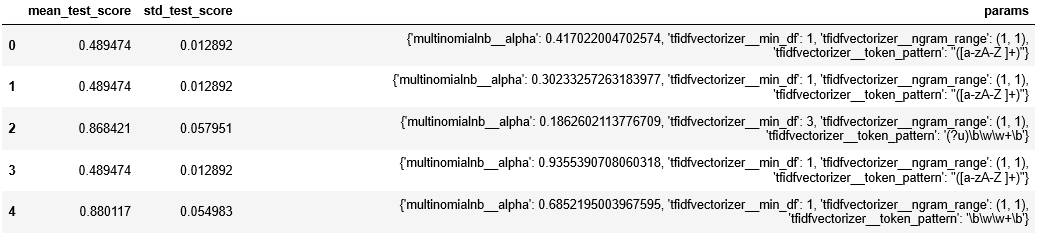
**Table 4**. *Models performance comparison for Sentiment task using text feature only – Part 1*

|  |  |  |
| --- | --- | --- |
| *text feature* | 1. BernoulliNB with CountVectorizer(binary = true) | 2. MultinomialNB with CountVectorizer (min\_df=5) |
| Training | 71.21 % | 77.03 % |
| Testing | 86.96 % | 86.96 % |
| Confusion Matrix |  |  |
| Report |  |  |

**Table 5**. *Models performance comparison for Sentiment task using text feature only – Part 2 and RandomizedSearchCV*

|  |  |  |
| --- | --- | --- |
| *text feature* | 3. MultinomialNB with TfidfVectorizer (min\_df=5) | RandomizedSearchCV |
| Training | 68.13 % | MultinomialNB with TfidfVectorizer  Random Search Best Score:  88.01 %  Random Search Best Parameters:  {'multinomialnb\_\_alpha': 0.6852195003967595,  'tfidfvectorizer\_\_min\_df': 1,  'tfidfvectorizer\_\_ngram\_range': (1, 1),  'tfidfvectorizer\_\_token\_pattern': '\\b\\w\\w+\\b'} |
| Testing | 91.3 % |
| Confusion Matrix |  |
| Report |  |

**Figure 5.** *Grid search result for Sentiment task using text feature only*



The advanced approach, combining manual features with document term matrix, was applied as well. The manual feature that was included is ‘length’ based on the observation on Figure 3. ‘exclamat’ feature was not included because it was performing worse than the single feature, ‘length’. In the advanced approach, the best candidate model was the second model, MultinomialNB with CountVectorizer. It yielded the accuracy of 82.61% and the overall performance was better than the first model, BernoulliNB with CountVectorizer (binary = true). The re-trained version of the second model yielded the accuracy of 92.4% with the random grid search technique.

**Table 6**. *Models performance comparison for Sentiment task with manual feature – Part 1*

|  |  |  |
| --- | --- | --- |
| *manual ft* | 1. BernoulliNB with CountVectorizer(binary = true) | 2. MultinomialNB with CountVectorizer |
| Training | 66.81 % | 62.31 % |
| Testing | 82.61 % | 82.61 % |
| Confusion Matrix |  |  |
| Report |  |  |

**Table 7**. *Models performance comparison for Sentiment task with manual feature – Part 2 and RandomizedSearchCV*

|  |  |  |
| --- | --- | --- |
| *manual ft* | 3. MultinomialNB with TfidfVectorizer | RandomizedSearchCV |
| Training | 53.63 % | MultinomialNB with CountVectorizer  Random Search Best Score:  92.4 %  Random Search Best Parameters:  {'featureunion\_\_pipeline \_\_countvectorizer\_\_min\_df': 5,  'featureunion\_\_pipeline \_\_countvectorizer\_\_token\_pattern': '\\b\\w\\w+\\b',  'multinomialnb\_\_alpha':  0.9325573593386588} |
| Testing | 39.13 % |
| Confusion Matrix |  |
| Report |  |

**Figure 6.** *Grid search result for Sentiment task with manual feature*

A screenshot of a cell phone

Description automatically generated

Authenticity  
Three combination of models were used throughout the training process for the lie detection task as well. The first model was used BernoulliNB algorithm and CountVectorizer with the setting, ‘binary=true’. The second model was built on MultinomialNB with CountVectorizer. The third model was utilized the same algorithm, MultinomialNB, to build the model with TfidfVectorizer. The detail information was shown in Table 8 and Table 9. Among the three models, the first model, BernoulliNB with CountVectorizer (binary = true), was performing the best with 43.48% on the test set. The random grid search result was included in the Table 9 showing that the re-trained model yielded the accuracy of 59.94% with entire dataset. The overall performance is very well but it is satisfying.

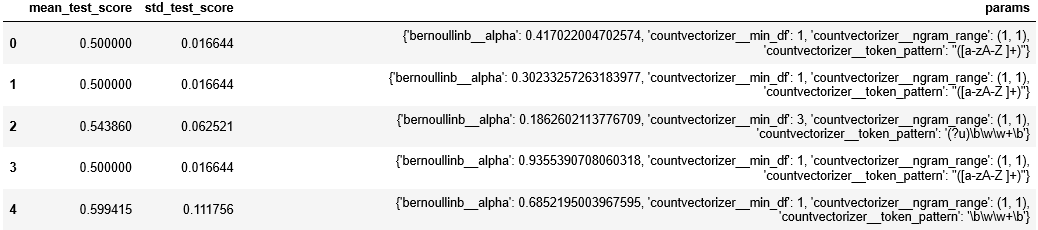
**Table 8**. *Models performance comparison for Lie task using text feature only – Part 1*

|  |  |  |
| --- | --- | --- |
|  | 1. BernoulliNB with CountVectorizer (binary = true) | 2. MultinomialNB with CountVectorizer |
| Training | 53.63 % | 55.16 % |
| Testing | 43.48 % | 34.78 % |
| Confusion Matrix |  |  |
| Report |  |  |

**Table 9**. *Models performance comparison for Lie task using text feature only – Part 2 and RandomizedSearchCV*

|  |  |  |
| --- | --- | --- |
|  | 3. MultinomialNB with TfidfVectorizer (min\_df=5) | RandomizedSearchCV |
| Training | 52.31 % | BernoulliNB with CountVectorizer (binary = true)  Random Search Best Score:  59.94 %  Random Search Best Parameters:  {'bernoullinb\_\_alpha': 0.6852195003967595,  'countvectorizer\_\_min\_df': 1,  'countvectorizer\_\_ngram\_range': (1, 1),  'countvectorizer\_\_token\_pattern': '\\b\\w\\w+\\b'} |
| Testing | 34.78 % |
| Confusion Matrix |  |
| Report |  |

**Figure 6.** *Grid search result for Lie task using text feature only*



With the advanced approach, combining manual features with document term matrix, the manual features, ‘exclamat’ and ‘neg\_long\_len’, were included to build models. In this approach, the best candidate model was the first model, BernoulliNB with CountVectorizer (binary = true). It yielded the accuracy of 43.48% and the overall performance was better than others. The re-trained version of the first model yielded the accuracy of 61.87% with the random grid search technique. The model shows the improvement with the added manual features.

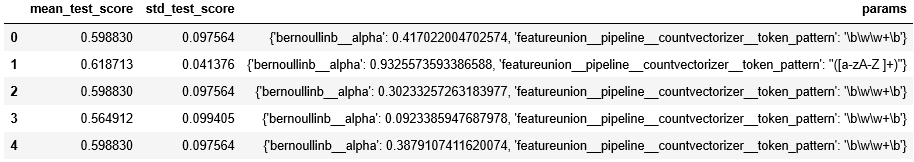
**Table 10**. *Models performance comparison for Sentiment task with manual feature – Part 1*

|  |  |  |
| --- | --- | --- |
| *manual ft* | 1. BernoulliNB with CountVectorizer(binary = true) | 2. MultinomialNB with CountVectorizer |
| Training | 53.63 % | 56.7 % |
| Testing | 43.48 % | 34.78 % |
| Confusion Matrix |  |  |
| Report | A screenshot of a cell phone  Description automatically generated | A screenshot of a cell phone  Description automatically generated |

**Table 11**. *Models performance comparison for Sentiment task with manual feature – Part 2 and RandomizedSearchCV*

|  |  |  |
| --- | --- | --- |
| *manual ft* | 3. MultinomialNB with TfidfVectorizer | RandomizedSearchCV |
| Training | 55.16 % | BernoulliNB with CountVectorizer (binary = true)  Random Search Best Score:  61.87 %  Random Search Best Parameters:  {'bernoullinb\_\_alpha': 0.9325573593386588,  'featureunion\_\_pipeline\_\_countvectorizer  \_\_token\_pattern': "'([a-zA-Z ]+)'"} |
| Testing | 39.13 % |
| Confusion Matrix |  |
| Report | A screenshot of a cell phone  Description automatically generated |

**Figure 7.** *Grid search result for Lie task with manual feature*



# Results

After going through the model training and model selection process. The best models for both of the tasks were shown as the Table 12 below. With manual feature added in the Sentiment task, the performance was increased by 4%. However, by having two manual features in the Authenticity task, the performance was increased only 2%. By comparing both models, it is clear to see that the lie detection task is a hard task to detect the intended result.

Although the lie detection performance is not persuasive. There are some reasons could explain why. First, the best model was not good enough even it did yield the satisfying result but still have room for improvement. Second, the size of the data was not big enough for an algorithm to learn the pattern of writing a fake review. Third, there are other factors need to be considered when one wants to perform the fake review detection such as time writing a review, gender, places…etc. Last, the ‘lie’ label could be not the ground truth. With the reasons mentioned above, one can say that the model did not learn the pattern very well. Other techniques should be considered for doing the Authenticity prediction task.

**Table 12**. *Best model for both of tasks*

|  |  |  |
| --- | --- | --- |
|  | Sentiment | Authenticity |
| Traditional Text Model | |  | | --- | | **Model** | | MultinomialNB with TfidfVectorizer | | Random Search Best Score**:** 88.01% | | |  | | --- | | **Model** | | BernoulliNB with CountVectorizer (binary) | | Random Search Best Score**:** 59.94% | |
| Traditional Text Model with Manual Features | |  | | --- | | **Model** | | MultinomialNB with TfidfVectorizer | | **Manual Features** | | ‘length’ | | Random Search Best Score**:** 92.4 % | | |  | | --- | | **Model** | | BernoulliNB with CountVectorizer (binary) | | **Manual Features** | | 'neg\_long\_len', 'exclamat' | | Random Search Best Score**:** 61.87% | |

# Conclusion

Reviews can be a tool or an indicator for people to decide whether buy a product or not, whether go to a restaurant for eating or not. People tend to rely on reviews more and more. To adapt the trend of relying on the reviews, many companies start to provide the review function for customers to leave their thought and feeling about a product, a transaction, or a shopping experience. It not only provides the evidence that their product/service is good enough for people to leave a comment on their system as a recommendation but also create an opportunity that their customer service can resolve customers’ issue.

However, reviews can be deceiving. When it comes to business, companies will try to find a way to compete their competitors. In that case, some companies may take the unethical route to undermine others’ reputation. And, that hurts the company who are working their business normally and legally. With the technique of Authenticity detection, it helps those company come up smelling of roses. Therefore, the better Authenticity detection can balance the fairness of the market, disclose the truth to the public and unveil the harmful intentions.

# Reference

MultinomialNB, from scikit-learn [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) ; 2 BernoulliNB, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html?highlight=nb#sklearn.naive_bayes.BernoulliNB)  
3 CountVectorizer, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html) ; 4 TfidfVectorizer, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)  
5 FeatureUnion, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.FeatureUnion.html) ; 6 make\_union, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.make_union.html)  
7 FunctionTransformer, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.FunctionTransformer.html?highlight=transformer#sklearn.preprocessing.FunctionTransformer) ; 8 cross\_val\_score, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html?highlight=cross_val#sklearn.model_selection.cross_val_score)  
9 Pipeline, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline) ; 10 make\_pipeline, from scikit-learn. [Web Link](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.make_pipeline.html?highlight=make_pipeline#sklearn.pipeline.make_pipeline)