Language Classifier for Tweets

Tianjin Ji

Project Final Report

Feb 7th, 2024

Abstract

In this project, we use machine learning techniques to classify if a given tweet is English or not for the UMass Global English on Twitter dataset. The machine learning models applied to the dataset are KNN, support vector machine, logistic regression, random forest, and gradient boosting. We perform cross validation on the training dataset to find the optimal parameters for each model. After fitting the models with the optimal parameter, we compute accuracy, precision, recall and f1-score for the testing dataset. The five classification models show good predictions (around 90% accuracy rate) for "definitely English" and "definitely not English" group, but poor prediction for "Ambiguous" group. This indicates that machine learning methods could be applied to classify "definitely English" and "definitely not English" categories, but manual classification is needed to correctly classify the "Ambiguous" group. In addition, computational challenges related to high-dimensional data are also discussed in this project.

Introduction

Extremely large amount of unorganized text data is generated on social media daily. However, the challenge of natural language processing includes annotating and analyzing these messy and hard-to-understand text data. On the other hand, businesses and social media companies are eager to find the pattern and meaning of the social media text data to help them better understand the needs of consumer and adjust their strategies accordingly. Data mining and machine learning can help us begin to understand these text data and find potential patterns and trends. A language classifier could be the first step to help us understand the meaning of messy tweets so we can effectively analyze these data. After we understand the content of the tweets, we can use them to identify spam, send automatic messages and for ads recommendation. Another advantage of building a language classifier is that people prefer to read and write in a language they understand. Since English is a common language, the first step of a language classifier could be to identify whether a given text is English or not. To demonstrate the use of machine learning techniques to build a language classifier, we analyze a pre-annotated small dataset to determine if a given tweet is English or not in this project.[1] We use some of the common classification machine learning techniques, namely, KNN, SVM, Logistic Regression, random forest and gradient boosting for classification.

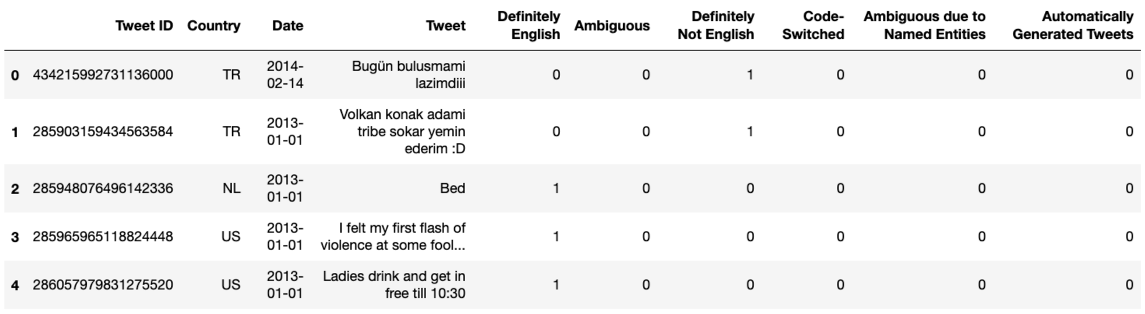
Data mining challenge for building a language classifier for social media data lies in the accurate pre-processing and annotation of text data. The dataset we choose is already pre-cleaned and annotated by researchers so we can trust the validity of the data. To build a language classifier using machine learning model, we need to have the correct annotation for each text string about which language it belongs to. This task could be challenging as manual annotation could be time consuming for large amount of data.

In this project, we first pre-process the data and extract the relevant columns. Then we perform exploratory data analysis to detect if there is any outliers or interesting trend. Next, we split the data into training dataset and testing dataset and perform feature extractions on the text data. To improve the performance of the models, we then perform dimension reduction on the data. Finally, we fit the data with the selected dimensions and compute test accuracy for different classification models using Python scikit-learn package.[2] We also compare the computation time for different models since some of them are computationally expensive. Lastly, we compare model performance for the testing data regarding precision, recall and recall for the three different categories and analyze the differences.

Data Source and Exploratory Data Analysis

The dataset is the UMass Global English on Twitter dataset on Kaggle.[3] This dataset consists of 10,502 tweets from 130 countries from 2013 and 2014. Table 1 shows the first five rows for this dataset. In this dataset, we will extract “Country”, “Tweet”, “Definitely English”. “Ambiguous” and “Definitely Not English” columns. The “Definitely English”. “Ambiguous” and “Definitely Not English” columns are the pre-annotated by researchers to identify if a given tweet in the “Tweet” column is English or not and these categories are mutually exclusive. The “Country” column is also selected because it represents the country where the tweet is sent and might be beneficial to classify the language of a given tweet. The rest of the columns are included in the analysis because they have irrelevant information (Tweet ID provides little value to predict the language of the tweet) or have too many empty or zero values (for columns such as “Code-Switched”, “Ambiguous due to Named Entities” and “Automatically Generated Tweets”).

Table 1. First five rows of the Twitter dataset



	Tweet ID	Country	Date	Tweet	Definitely English	Ambiguous	Definitely Not English	Code-Switched	Ambiguous due to Named Entities	Automatically Generated Tweets
0	434215992731136000	TR	2014-02-14	Bugün bulusmami lazimdiii	0	0	1	0	0	0
1	285903159434563584	TR	2013-01-01	Volkan konak adami tribe sokar yemin ederim :D	0	0	1	0	0	0
2	285948076496142336	NL	2013-01-01	Bed	1	0	0	0	0	0
3	285965965118824448	US	2013-01-01	I felt my first flash of violence at some fool...	1	0	0	0	0	0
4	286057979831275520	US	2013-01-01	Ladies drink and get in free till 10:30	1	0	0	0	0	0

The problem statement for this project is to use “Country” and “Tweet” columns to predict the response variable y(1 ~ 'Definitely English’ , 2~ 'Ambiguous’, 0 ~ 'Definitely Not English’). None of the tweets only contains symbols or punctuation marks only. For data preprocessing, We first remove nan rows since there are only 10 of them. We then combine response variable y into one column (1 ~ 'Definitely English’ , 2~ 'Ambiguous’, 0 ~ 'Definitely Not English’) for easier model fitting and prediction since these categories are mutually exclusive. Lastly, We encode Country into numerical value using LabelEncoder method in sklearn from Python, this feature is scaled later before dimension reduction and model fitting.[4]

For exploratory data analysis, we first plot the distribution of the response variable y. Figure 1 shows this distribution. Figure 1 shows that that less than slightly half of the tweets are Definitely not English and around half of the tweets are Definitely English, and the Ambiguous category only has around 7% of the data.

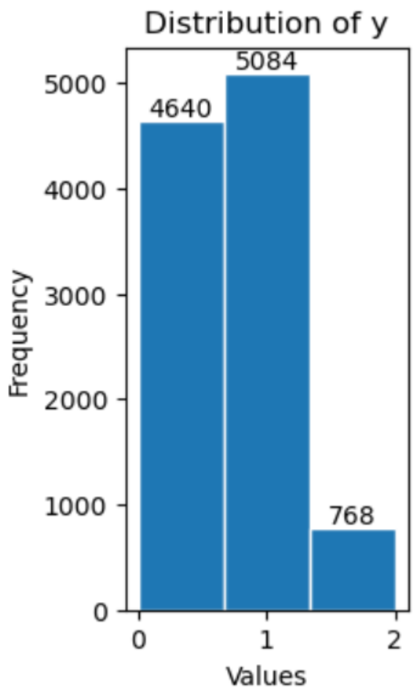


Figure 1. Distribution for the response variable y

Because the dataset consists of 130 countries, we plot the top 10 countries with the highest tweets count, which is shown in Figure 2 and Table 2 below. Figure 2 and Table 2 show that the top country is the United States, with nearly all tweet classified as Definitely English; Britain has lots of Tweets that are not English and other non-English speaking countries such as Japan has majority of tweets that are not English; The ambiguous group has tweets that are not in English-speaking countries, making it hard to interpret. The plot to show the tweets count for all countries are shown in the Appendix section. The exploratory data analysis shows that there is no obvious outlier for this dataset, this is mainly because this dataset is pre-annotated, and tweets are already selected by researchers.[1]

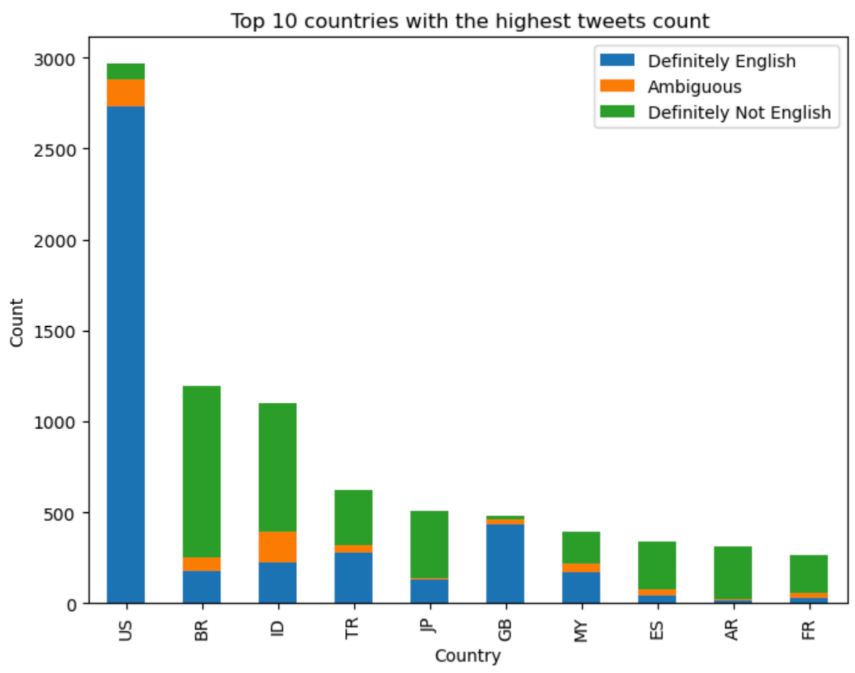
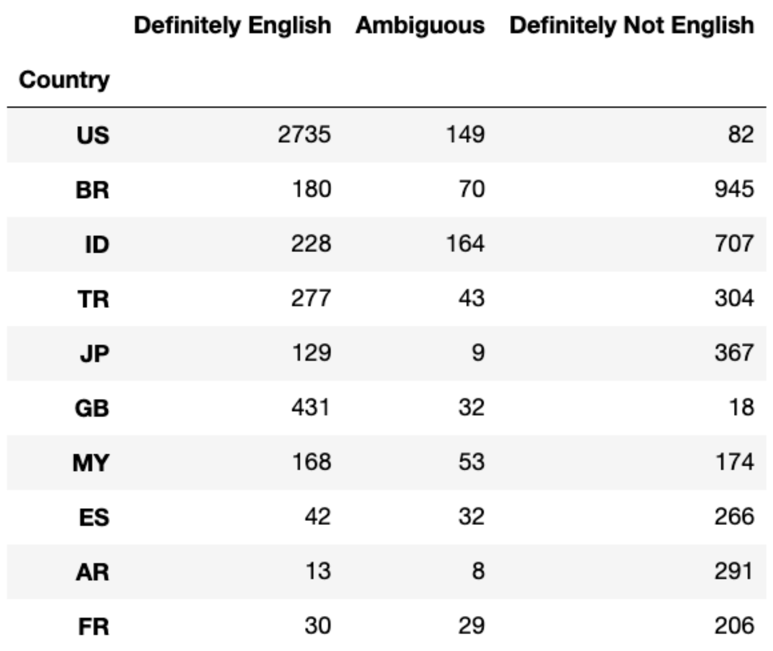


Figure 2. Top 10 countries with the highest tweets count bar plot

Table 2. Top 10 countries with the highest tweets count



Country	Definitely English	Ambiguous	Definitely Not English
US	2735	149	82
BR	180	70	945
ID	228	164	707
TR	277	43	304
JP	129	9	367
GB	431	32	18
MY	168	53	174
ES	42	32	266
AR	13	8	291
FR	30	29	206

Proposed Methodology

Figure 3 shows a flow chart for the proposed methodology. To analyze the data, we first split the data into 80/20 split, resulting in 8393 training data points and 2099 testing data points. We then perform feature extraction on the tweets text data for the “Tweet” column. Since this results in over 16,000 dimensions, we perform feature selection/dimension reduction use SVD(singular value decomposition). We mainly use Python sklearn package for model fitting and parameter tuning.[2] We first fit the selected features into our classification models (KNN, SVM, logistic regression, random forest, and gradient boosting) and then use cross validation to find the best parameter. These classification methods are chosen because KNN, SVM and logistic regression are common classification techniques and random forest, and gradient boosting are common ensemble learners. Gradient boosting is chosen because we are interested in if boosting can improve the performance for random forest. We choose either 10 or 20 folds for CV depends on the model computational times. The average test accuracy is used as a metrics to select the optimal parameter. To evaluate model performance, we fit the model with optimal parameter and compare performance for the testing data and calculate the accuracy, precision, recall and f1-score. Because high-dimension data analysis takes a long time, including dimension reduction, model fitting and tuning, we are not performing Monte Carlo CV on the testing dataset to obtain the average accuracy rate for the models due to limitation of computational power. The other reason is that we have 2099 data points for the testing dataset, which is not small. The limitation of our analysis includes the limitation of our computational power. With the expansion of computational power, future work could include testing these models on using 100 iterations of Monte Carlo CV.

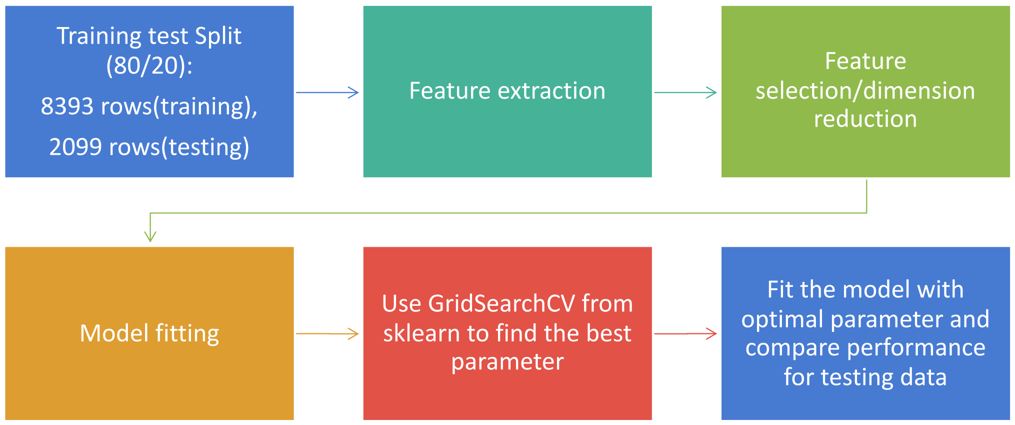

Training test Split (80/20):
8393 rows(training),
2099 rows(testing)
Feature extraction
Feature selection/dimension reduction
Model fitting
Use GridSearchCV from sklearn to find the best parameter
Fit the model with optimal parameter and compare performance for testing data

Figure 3. Flow chart for the proposed methodology

Analysis and Results

The technical detail for each subsection is included in the Appendix. The following results are generated using Python sklearn packages.[2]

Dimension reduction

After feature extractions of the text data (details in Appendix), we have 16,329 columns. Since we have too many features, over 16,000, we perform dimension reduction so our models can run faster.[6] After dimension reduction we have successfully reduced these features to around 3000, which means that more than 80% of the features are reduced.

KNN

For KNN, we have very fast computational time per run of the model, around 0.1-0.2 sec. This is probably because it's a non-parametric model. Using 20-fold CV, we find that the optimal K is equal to 12 with training error of around 0.13(mean test accuracy of around 0.87), as shown in Figure 4. The dimension reduction has helped in saving computational time for KNN.

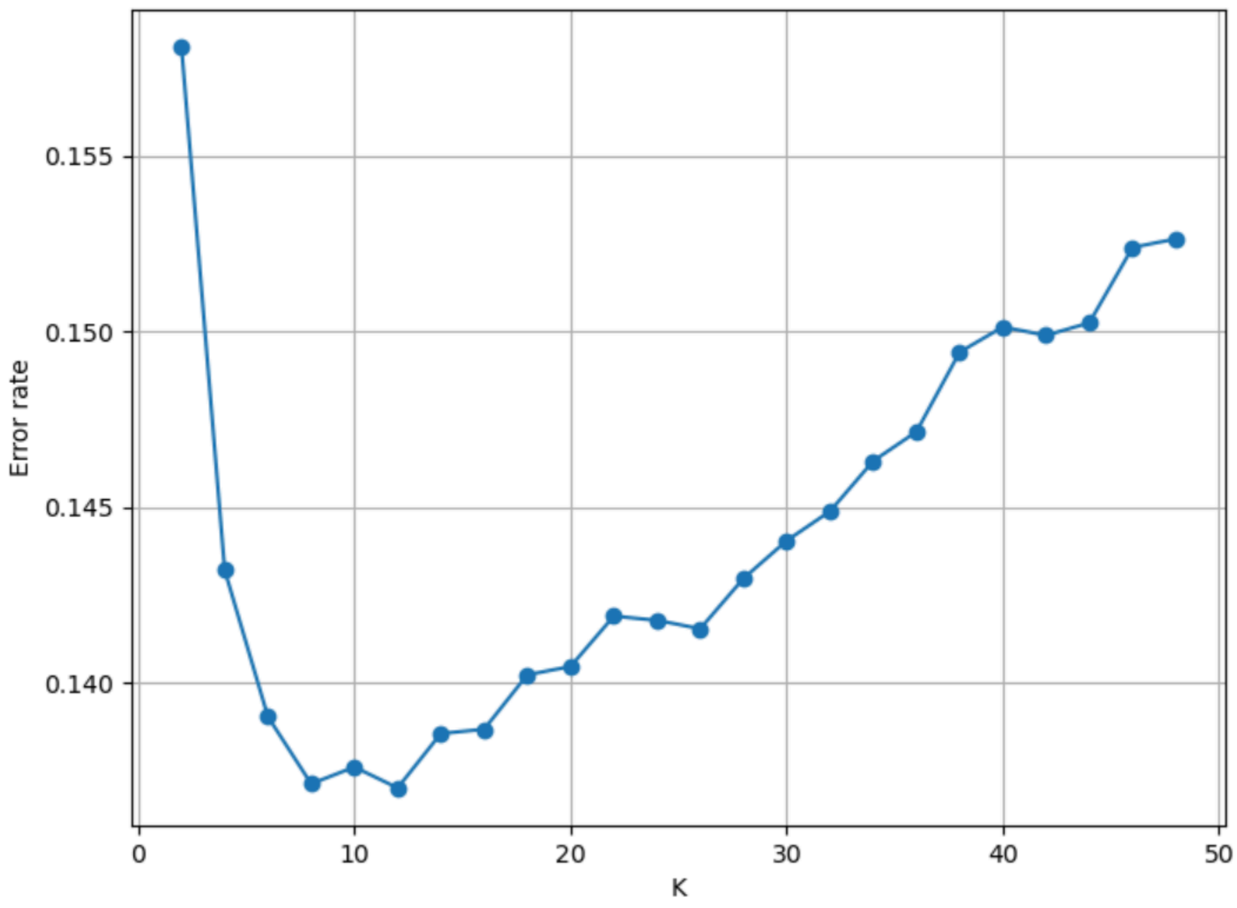


Figure 4. Elbow plot for 20-fold CV for KNN

SVM

The computational time for SVM is much greater than KNN, around 1-2 min per run depends on different parameter selection. Figure 5 shows the 10-fold CV results, with the optimal C of 10 and gamma of 0.1. This optimal parameter gives a mean test accuracy of around 0.87.

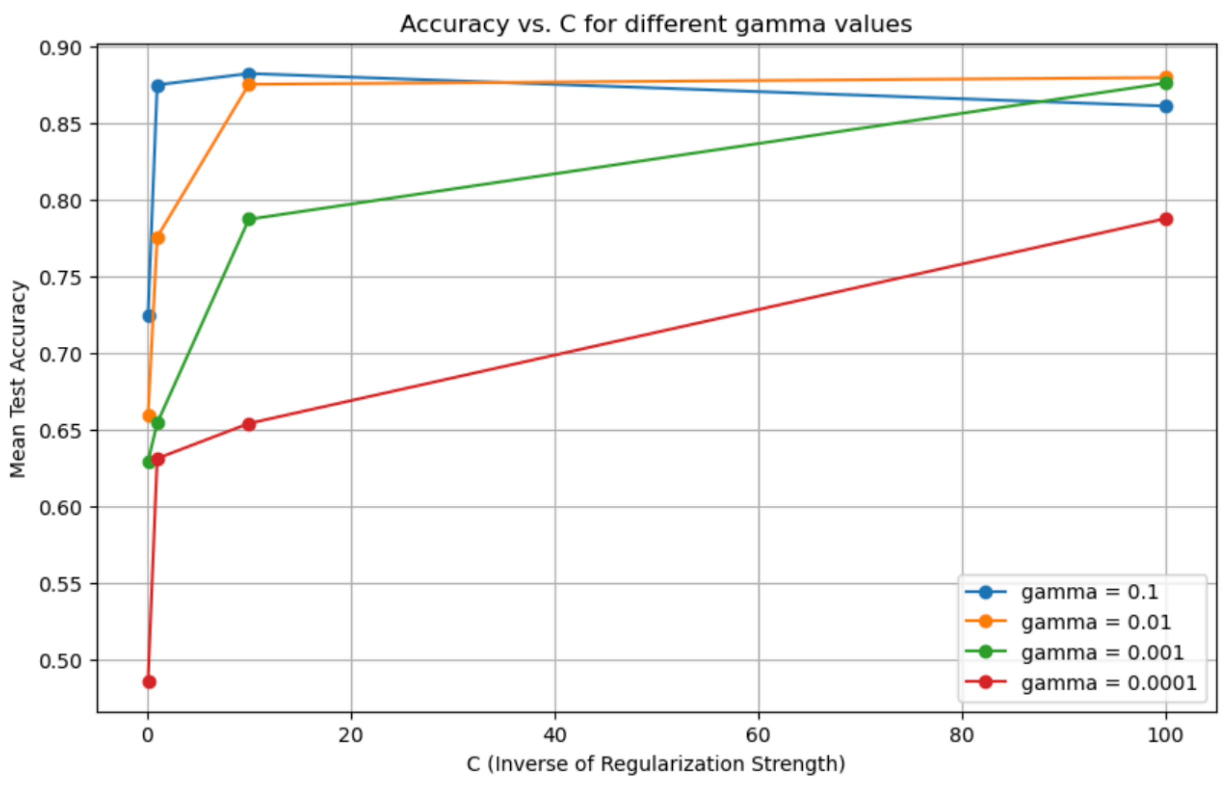


Figure 5. 10-fold CV result for SVM

Logistic Regression

Logistic regression has faster computational time than SVM. Depends on the choice of parameters, each run takes around 5 to 40 secs. Figure 6 shows the 20-fold CV result for logistic regression. This optimal parameter gives a mean test accuracy of around 0.88.

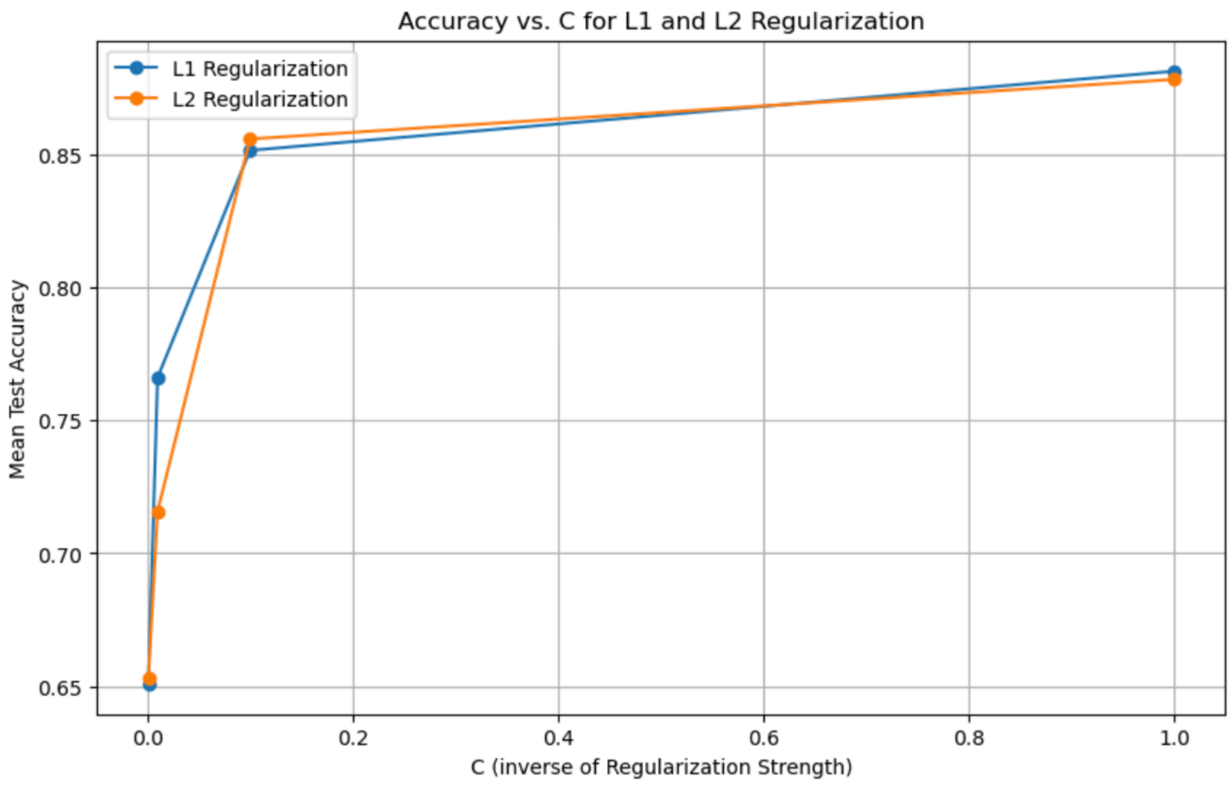


Figure 6. 20-fold CV result for logistic regression

Random forest

The computation time varies between 15 seconds and around 2min, mainly depends on the n_estimators, which is the number of trees in the forest. In general, we get a higher accuracy rate with larger number of estimators, with the tradeoff of higher running time. The 10-fold CV result plotted in Figure 7 shows that the optimal parameter is with n_estimators of 150 and max_features of 200 with a mean test accuracy of around 0.86.

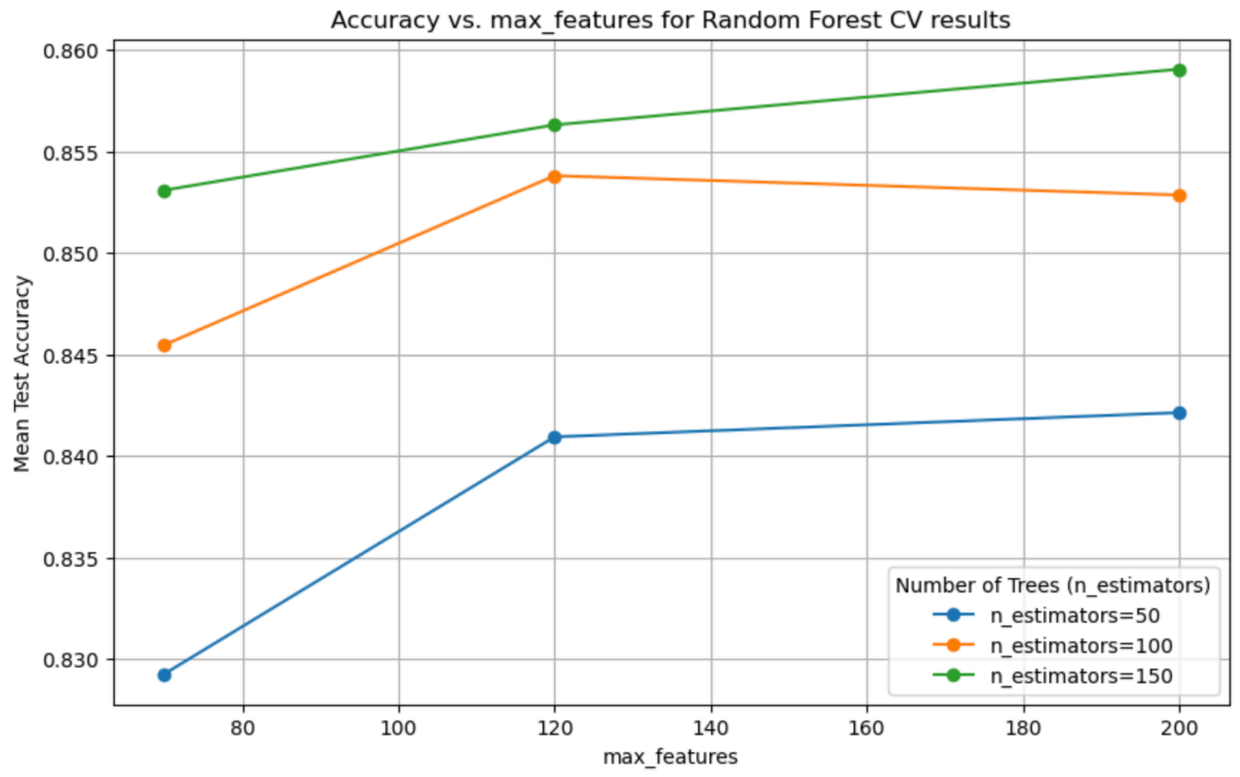


Figure 7. 10-fold CV result for random forest

Gradient boosting

We fit the gradient boosting classifier with sklearn.[11] This algorithm runs slightly longer than the random forest. Figure 8 shows the optimal parameter that gives the highest mean test accuracy (around 0.87) is n_estimator = 150, learning_rate = 0.2. As n_estimator becomes larger, the running time is also longer. Figure 8 also shows that there is not much significant difference when learning rate is 0.2 with n_estimators of 100 and 150.

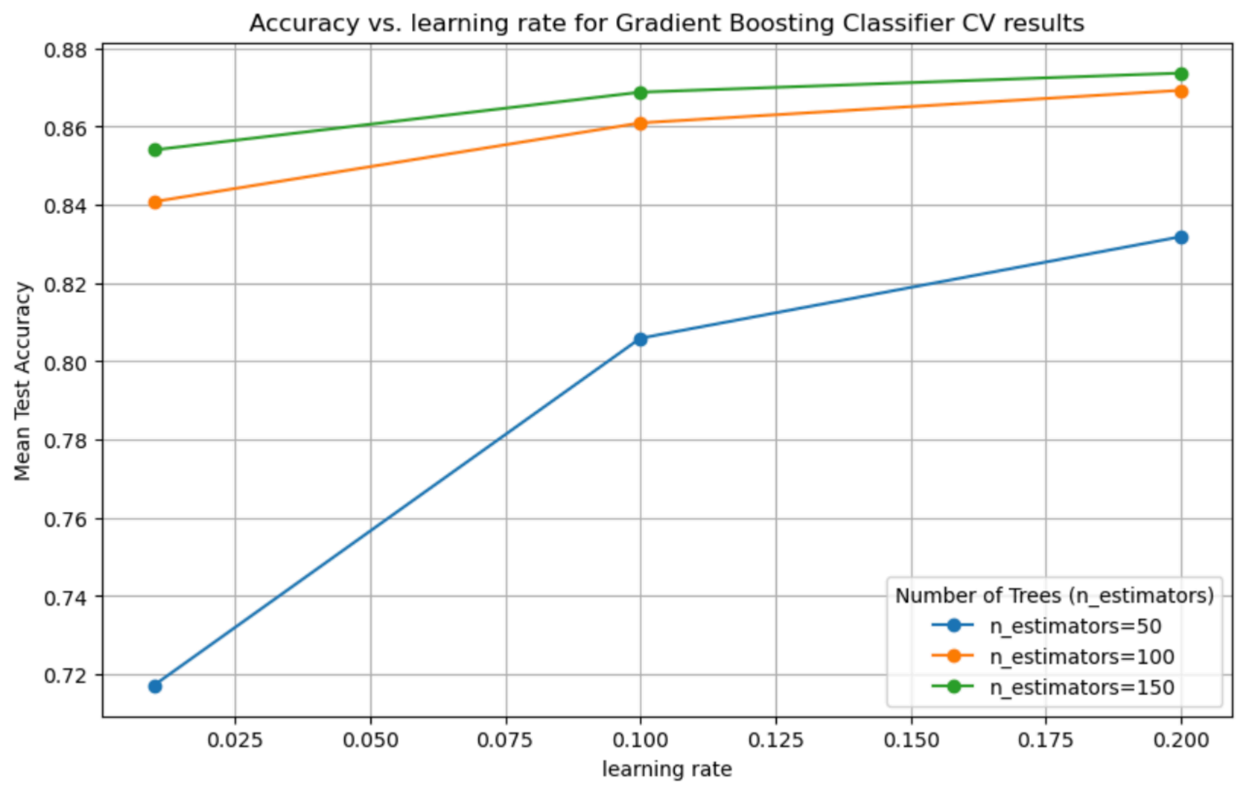


Figure 8. 10-fold CV result for gradient boosting

Testing dataset result

Table 3 shows the testing accuracy for the five models after fitting the models with the optimal parameters selected in cross validation. Table 3 shows that SVM and Logistic Regression performs the best with the highest accuracy. With regards to computational time , KNN runs fastest(within 0.2 sec) and logistic regression also runs very fast(within seconds). Although SVM has good test accuracy, it runs very slow(around 2 min) and might not work well for high-dimensional data. Random forest and gradient boosting have similar test accuracy although gradient boosting runs slightly longer than random forest. The running time for random forest and gradient boosting is around like that of SVM (around 1-2 min).

Table 3. Testing accuracy for five models

	KNN	SVM	Logistic Regression	Random Forest	Gradient Boosting
Testing accuracy	0.85	0.88	0.88	0.85	0.86

Since we have three different classification categories, we generate confusion matrix and compute compute the precision, recall and f-1 score for each category. Table 4 shows the summary statistics for the confusion matrix of the five models, including precision, recall and f1-score. Table 2 shows that category 2, which represents the “Ambiguous” group, has the lowest score for precision, recall and f1-score. For the 0(“Definitely not English”) and 1(Definitely English”) categories, the five models generate much better predictions than “Ambiguous” group. One thing to note is that SVM and Logistic regression has generated better metrics for precision, recall and f1-score especially for the Ambiguous group, as well as slightly better predictions for the other two groups. This finding could indicate that we need manual classification for the “Ambiguous” group because it’s hard for these machine learning model to perform accurate predictions for this category.

Random Forest fails to predict the “Ambiguous group”, resulting in 0 for the metrics. This is probably because fewer data belong to this group and the majority vote makes the classification go to the other two category. On the other hand, gradient boosting generates some prediction for the ambiguous group, so it’s an improvement of the random forest model.

Table 4. Precision, recall and f1-score for the five models.

KNN result

```
   precision    recall  f1-score

0       0.83      0.92      0.87
1       0.87      0.91      0.89
2       0.75      0.02      0.04
```

SVM result

```
   precision    recall  f1-score

0       0.88      0.93      0.90
1       0.91      0.93      0.92
2       0.47      0.23      0.31
```

Logistic Regression result

```
   precision    recall  f1-score

0       0.88      0.93      0.91
1       0.90      0.94      0.92
2       0.56      0.18      0.28
```

Random Forest result

```
   precision    recall  f1-score

0       0.85      0.90      0.87
1       0.85      0.94      0.89
2       0.00      0.00      0.00
```

Gradient Boosting result

```
   precision    recall  f1-score

0       0.83      0.94      0.88
1       0.89      0.91      0.90
2       0.75      0.08      0.14
```

Conclusions

In summary, the five classification models (KNN, SVM, Logistic regression, random forest, and gradient boosting) show good predictions (around 90% accuracy rate) for "definitely English" and "definitely not English" group, but poor prediction for "Ambiguous" group. This language classifier could be the first step in understanding and interpreting social media data. Our result also shows that machine learning isn't about fitting various models, we need human judgment in correctly classification the "Ambiguous" group.

The applications for this language classifier include building apps that help brand promote products using the language consumers speak and understand. Some of the ambiguous group classification could lead to identification of spam and phishing acts on social media websites. Because language is associated with cultural context, language classifier with different popular language filter could help identify trends among different ethnic groups. In addition, we could continue to expand our language classifier to classify multiple languages from social media sites and build applications that detect trends from these languages, which helps business understand consumers' needs better.

Our work is mainly limited by the computational power, and it brings up challenges for proper processing and analysis for high-dimensional data. Future work could include further study of non-parametric models such as KNN because it has computational advantage. Future work could also include using a large testing dataset or Monte Carlo CV for comparison between models.

Since Sklearn has limited computational power for large dataset, we can explore other machine learning models (neuron networks) from Pytorch or TensorFlow to compare the performance of other models. We could also try to explore better feature selection methods for NLP (e.g. Pytorch supports multiple language extractions) for multi-language classifier.

The potential challenge when processing large quantity of social media data is Annotation for millions of tweets could be costly. Human review is necessary to prevent junk in – junk out. To improve our predictions, future work could involve further analysis for the characteristic of "Ambiguous" group.

Lessons we have learned.

One of the most important lessons I learned from this course and this project is how to effectively tune the parameters for different machine learning methods. Another lesson I learned is that proper tuning using CV could computationally expensive and time-consuming. We need proper knowledge for the algorithms and relevant domain knowledge to select the best parameters to avoid issues such as overfitting. In addition, I learned from this project that it's challenging to process high-dimension data even with dimension reduction techniques. My dataset is relatively small, 1.3M, but it takes a very long time to perform dimension reduction and CV. I hope in the future, the course material could include some strategy to process high dimension data.

Appendix

EDA additional plot

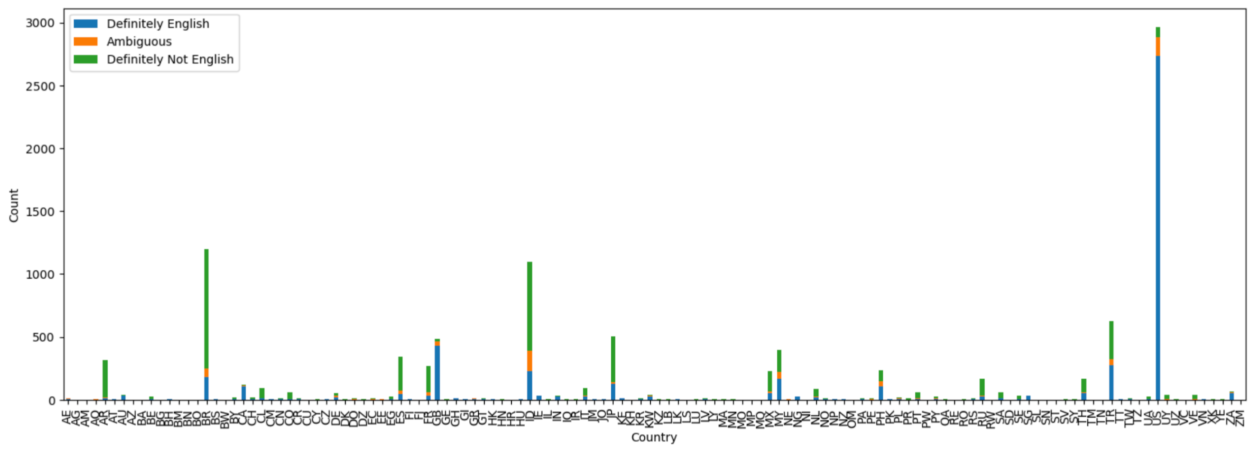


Figure 9. Count for Tweets for 130 countries

Feature extractions

For feature extraction of tweets, we use TfidfVectorizer from sklearn.[5] For this function, we set ngram_range=(1, 2) to obtain the frequency for 1 word or 2-word phrase, and the result is stored in a sparse matrix.[5] This parameter is selected because it's the most frequently used in NLP. If we expand the ngram_range, we will have a much larger features to perform any further analysis. Since we are limited by our computational power, we choose the ngram_range to be (1,2).

Dimension reduction

Because the data is stored in a sparse matrix, we use SVD (singular value decomposition) for dimension reduction.[6] We use TruncatedSVD() in Python scikit-learn for SVD, and we set the max_component to be 5,000.[6] This process is very time consuming and takes a while to run. Figure 4 shows the cumulative variance plot for the dimension reduction result. Figure 10 shows that the 99% cumulative explained variance (red dash line) is around 3,000 components, which means that more than 80% of the features are reduced. We then transform both training and testing data to the reduced 3,000 components.

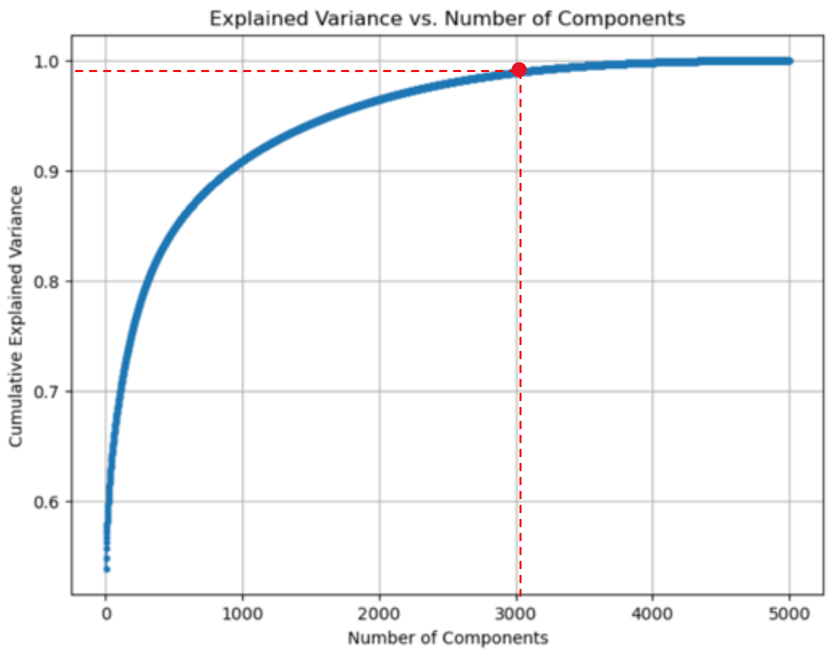


Figure 10. Cumulative Explained Variance plot for SVD

SVM

We use the rbf kernel for SVM in sklearn to fit SVM, which is similar to the Gaussian distribution.[7] We then tune the parameter C and gamma using 10-fold CV. According to sklearn documentation, C is the inverse of L2 regularization strength.[4] Smaller C means high regularization, larger margin between boundaries and less likely to overfit. On the other hand, gamma controls how far the influence of the data points: High gamma means nearby points have more influence over the decision boundary; Low gamma means faraway points have more influence over the decision boundary.[8]

Logistic Regression

Figure 6(in Analysis and Results) shows that the mean test accuracy differs every little when C≥0.1 with either L1 or L2 regularization. L1 regularization is like LASSO regression, which performs feature selection and L2 regularization is like ridge regression, which shrinks the coefficients but does not force them to be zero. Figure 6also shows that the optimal parameter is L1 regularization with C = 1, which corroborates L1 performs better with sparse matrix. This optimal parameter gives a mean test accuracy of around 0.88. During the tuning process, we encounter some computation difficulty with convergence for the sage solver for C≥10 so our CV tuned C from 0-1.

Random forest

We set the minimum leaf size to be 20 to prevent overfitting. We tune two parameters for 10-fold CV: n_estimators and max_features, which represents number of features to consider when looking for the best split.[10]

Gradient boosting

For the second ensemble method, we use the gradient boosting classifier with sklearn.[11] This ensemble method uses gradient boosting algorithm that uses additive model using forward stages.[11] N classes regression trees are fit on the negative gradient of the loss function for each stage to perform boosting.[11] For 10-fold CV, we set max_features=100, min_samples_leaf = 20 to save computational time. We tune n_estimators and learning_rate and the result is shown in Figure 8(in Analysis and Results).

Additional statistics for the confusion matrix of each model

KNN result

```
              precision    recall  f1-score   support

           0       0.83      0.92      0.87       934
           1       0.87      0.91      0.89      1007
           2       0.75      0.02      0.04       158

    accuracy                           0.85      2099
   macro avg       0.82      0.62      0.60      2099
weighted avg       0.84      0.85      0.82      2099
```

SVM result

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.90       934
           1       0.91      0.93      0.92      1007
           2       0.47      0.23      0.31       158

    accuracy                           0.88      2099
   macro avg       0.75      0.70      0.71      2099
weighted avg       0.86      0.88      0.87      2099
```

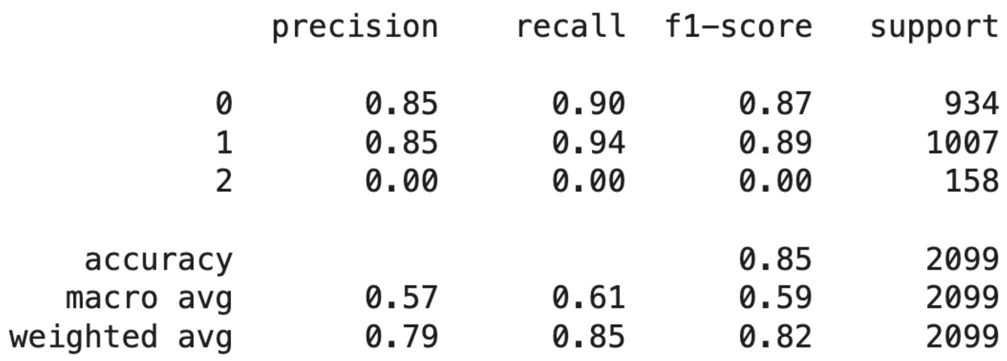
Logistic Regression result

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.91       934
           1       0.90      0.94      0.92      1007
           2       0.56      0.18      0.28       158

    accuracy                           0.88      2099
   macro avg       0.78      0.69      0.70      2099
weighted avg       0.86      0.88      0.86      2099
```

Random forest result*



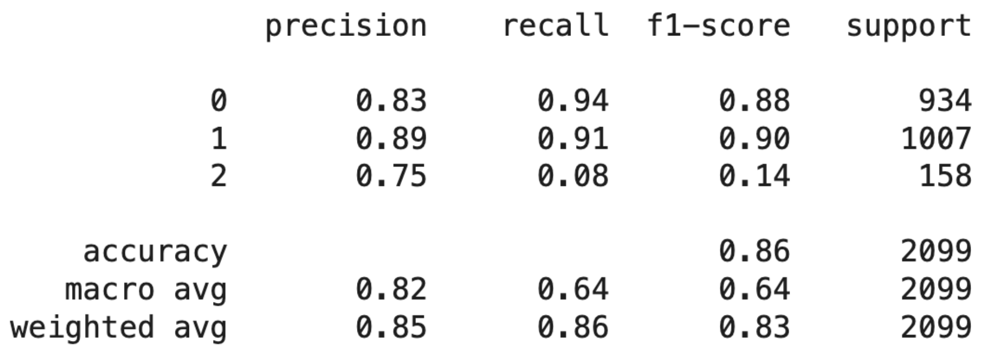
```
              precision    recall  f1-score   support

           0       0.85      0.90      0.87       934
           1       0.85      0.94      0.89      1007
           2       0.00      0.00      0.00       158

    accuracy                           0.85      2099
   macro avg       0.57      0.61      0.59      2099
weighted avg       0.79      0.85      0.82      2099
```

* Random forest has failed to predict for the “Ambiguous” group (category 2 in the above result

Gradient boosting result



```
              precision    recall  f1-score   support

           0       0.83      0.94      0.88       934
           1       0.89      0.91      0.90      1007
           2       0.75      0.08      0.14       158

    accuracy                           0.86      2099
   macro avg       0.82      0.64      0.64      2099
weighted avg       0.85      0.86      0.83      2099
```

References

1. A Dataset and Classifier for Recognizing Social Media English (https://aclanthology.org/W17-4408) (Blodgett et al., WNUT 2017)
2. Sklearn User Guide (https://scikit-learn.org/stable/user_guide.html)
3. The UMass Global English on Twitter Dataset (https://www.kaggle.com/datasets/rtatman/the-umass-global-english-on-twitter-dataset)
4. sklearn documentation - sklearn.preprocessing.LabelEncoder (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html)
5. sklearn documentation - sklearn.feature_extraction.text.TfidfVectorizer (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)
6. Singular Value Decomposition for Dimensionality Reduction in Python (https://machinelearningmastery.com/singular-value-decomposition-for-dimensionality-reduction-in-python/#:~:text=SVD%20is%20typically%20used%20on,to%20use%20the%20PCA%20method.)
7. sklearn documentation – sklearn.svm.SVC (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html)
8. sklearn documentation – RBF SVM parameters (https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html#:~:text=Intuitively%2C%20the%20gamma%20parameter%20defines,high%20values%20meaning%20'close')
9. sklearn documentation - sklearn.linear_model.LogisticRegression (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

10. sklearn documentation - sklearn.ensemble.RandomForestClassifier (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)

11. sklearn documentation - sklearn.ensemble.GradientBoostingClassifier (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html)