CSCI803 ASS2

Name: Yao Xiao SID: 2019180015

November 17, 2020

1 Sentiment Analysis

1.1 Describe the procedure of collecting the tweets and manually tagging them

We apply for consumer key and access token calls twitter API to get tweets related to Hate speech topics (like racist and sexism), and classifies related emotions with textblob, converts the data into correct input format, and classifies users' views into positive (non racist) and negative (racist) by tweeting. The specific algorithm is as follows:

```
import tweepy
from textblob import TextBlob

api = tweepy.API(auth
public_tweets = api.search('hate',lang="en")
for tweet in public_tweets:
  print(tweet.text)
  analysis = TextBlob(tweet.text)
  print(analysis.sentiment)
  if analysis.sentiment[0]>0:
    print("positive")
  elif analysis.sentiment[0]<0:
    print("negative")
  else:
    print("neutral")
```

1.2 Describe the statistics of the obtained data set

From the obtained the data, we can see the shape of train data is (31962, 3), the shape of test data is (10655, 2).

And the train data has two label, 0 and 1. Among them, the size of label-0 is 29720, and the size of label-1 is 2242, the data format is as follows:

```
In [39]: train_data[:10]
```

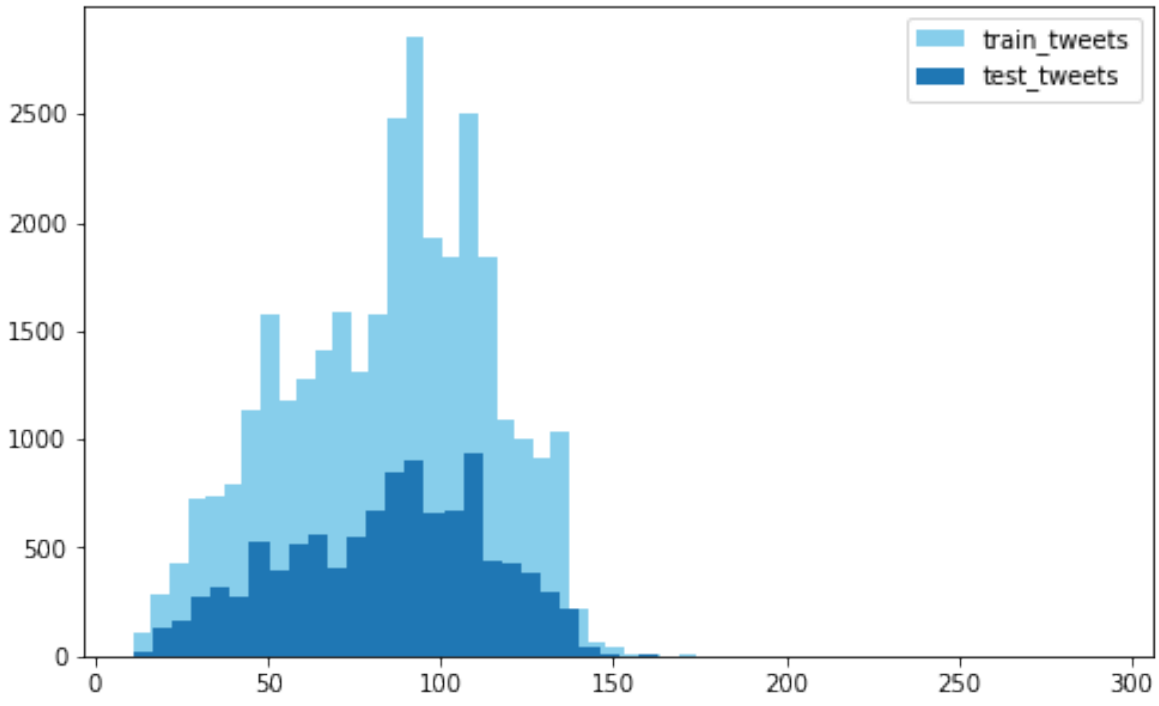
Out[39]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in urð ±!!! ð ð ð ð ð ¦ð ¦ð ¦
4	5	0	factsguide: society now #motivation
5	6	0	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo
6	7	0	@user camping tomorrow @user @user @user @user @user @user @user dannyâ€¦
7	8	0	the next school year is the year for exams.ð ¯can't think about that ð #school #exams #hate #imagine #actorslife #revolutionschool #girl
8	9	0	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers â€¦
9	10	0	@user @user welcome here ! i'm it's so #gr8 !

```
In [29]: train_data['label'].value_counts()
```

```
Out[29]: 0    29720
         1     2242
         Name: label, dtype: int64
```

Here is the simple analysis:



1.3 Describe how you represent each tweet for classification

In the beginning, we clear the data and get the pure text for further processing. For the nlp task, we first should cut a whole sentence into characters or words, also named toeknizing; and then convert

text to index.

Specific to the project, we created a function to create a vector for each tweet by taking the average of the vectors of the words present in the tweet, and then build the embedding matrix.

And for the pretrained, we choose Word2Vec model.

	0	1	2	3	4	5	6	7	8	9	...	190	191	192	19
0	0.123130	-0.022254	0.066942	-0.187952	0.204866	0.015465	0.162484	0.357025	-0.111437	-0.059919	...	-0.026463	-0.159017	0.042293	0.02467
1	-0.052064	0.086028	0.029135	-0.025141	0.112335	0.088224	0.073418	0.163126	-0.046456	-0.131191	...	-0.151946	-0.170070	-0.063647	0.05393
2	0.518164	-0.054258	-0.087906	0.036475	0.223916	-0.110665	0.127246	0.426922	0.280709	-0.426310	...	0.085781	0.108380	0.242764	0.27907
3	-0.103176	0.155302	0.074282	-0.132084	0.243130	0.187081	0.094765	0.304369	0.016580	-0.062620	...	-0.256293	-0.035999	-0.224414	0.26431
4	0.018459	0.060308	0.228547	0.347049	0.110722	0.064820	0.167679	0.544997	0.199003	-0.068122	...	-0.281098	-0.389499	-0.103064	0.29530
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
42612	0.005935	-0.060872	-0.256659	0.040887	0.222693	0.050371	-0.178915	0.426263	0.048664	-0.393445	...	-0.044317	0.053470	0.056523	-0.04607
42613	-0.130714	-0.085445	-0.049283	-0.305386	0.338653	-0.341602	-0.001439	0.445900	-0.027019	-0.152537	...	-0.096971	-0.225523	0.154751	0.15071
42614	0.091943	0.079687	-0.104432	-0.003061	0.152342	-0.021984	-0.040846	0.302879	-0.073467	0.011795	...	0.002968	-0.218642	0.008003	0.02376
42615	0.158446	0.238522	0.115380	0.002704	0.011900	-0.118903	-0.059619	0.322845	-0.109239	-0.267277	...	0.076083	0.106787	0.067033	0.21084
42616	-0.064486	-0.021902	0.002809	-0.019143	-0.084407	-0.102521	-0.025717	0.316613	0.177862	-0.062883	...	0.165235	-0.178788	0.077090	0.14894

1.4 For each classifier, describe its working principle, classification procedure, and parameter setting

1.4.1 Logistic Regression

Logistic Regression is a classification algorithm could help us predict whether the given Tweet is Racist or not. Logistic regression predictions are distinct. We can also view probability scores underlying the model's classifications. Logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

1.4.2 Naive Bayes

Naive Bayes is a classification technique based on Bayes' theorem, assuming independence between predictors. Bayes' theorem provides an equation to describe the relationship between conditional probabilities of statistics. In Bayesian classification, we are interested in the probability of finding a label given certain observation characteristics. There are three types of Naive Bayesian models. Gauss, polynomial and Bernoulli. Gaussian is basically used for classification problems, while polynomials are used to obtain different counts. If the feature vector is binary, Bernoulli is used.

1.4.3 SVM

SVM is a supervised machine learning algorithm for classification and regression problems. In SVM, we perform classification by finding a hyperplane that can distinguish two categories well. Kernel tricks are used to transform data and then find the best boundary between possible outputs based on the data. Hyperparameters such as kernel, regularization, gamma, and margins can be used to optimize SVM.

Settings as follows:

```
1        svc = svm.SVC(kernel='linear', C=1, probability=True)
```

1.4.4 Random Forest

Random forest is a supervised learning algorithm. It can be used for classification and regression. It is also the most flexible and easy to use algorithm. The forest is composed of trees. It is said that the more trees, the stronger the forest. Random forest creates decision trees on randomly selected data samples, obtains predictions from each tree and selects the best solution through voting.

Settings as follows:

```
rf = RandomForestClassifier(n_estimators=400, random_state=12)
```

1.4.5 XGBoost

XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" originates from the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman.

$$\text{Prediction:} \quad \hat{y}_i^t = \hat{y}_i^{t-1} + f_t(x_i) \tag{1}$$

$$\text{Loss:} \quad \sum_{i=1}^{n} \ell(y_i, \hat{y}^{t-1} + f_t(x_i)) + \Omega(f_t) \tag{2}$$

The working process of XGBoost is that it combines predictions from multiple decision trees. All weak learners in the gradient boosting machine are decision trees. The trees in XGBoost are constructed sequentially, trying to correct the errors of the previous tree.

Settings as follows:

```
xgb = XGBClassifier(max_depth=6, n_estimators=1000, nthread= 3)
```

1.5 For each classifier, report the classification accuracy, AUC, and plot the confusion matrix

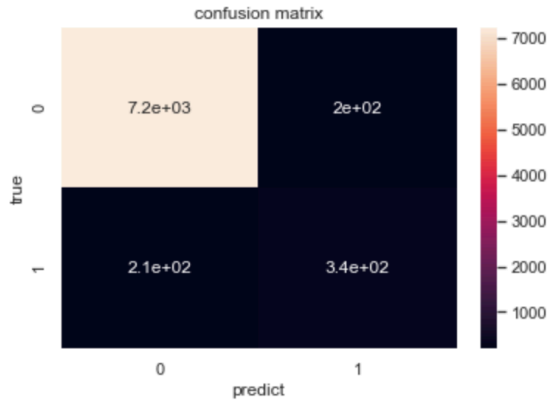
1.5.1 Logistic Regression

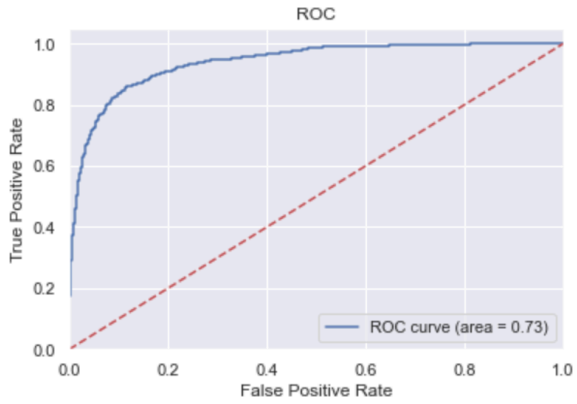
```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      7432
           1       0.64      0.62      0.63       559

    accuracy                           0.95      7991
   macro avg       0.80      0.80      0.80      7991
weighted avg       0.95      0.95      0.95      7991


Accuracy Score:  0.9486922788136654
F1 Score:  0.6272727272727273
Cross Validation Error:  0.2168262380533533
```





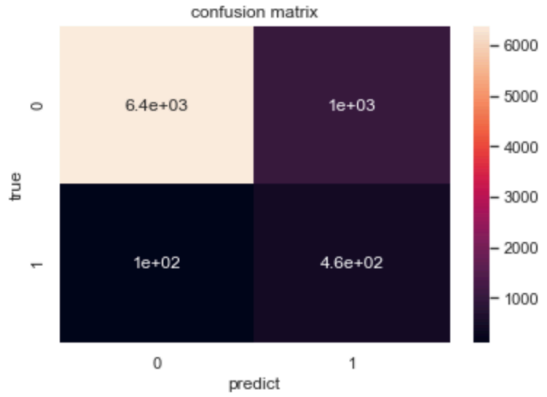
1.5.2 Naive Bayes

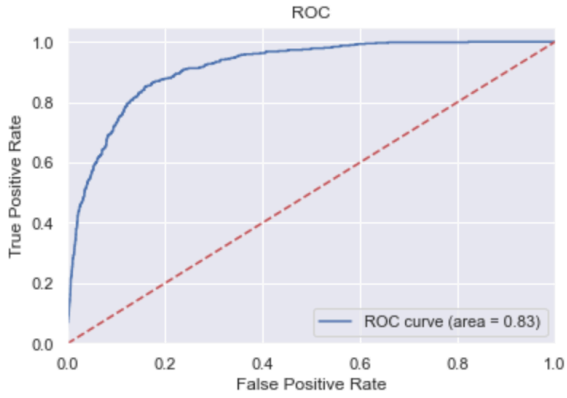
```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.86      0.92      7432
           1       0.30      0.82      0.44       559

    accuracy                           0.86      7991
   macro avg       0.64      0.84      0.68      7991
weighted avg       0.94      0.86      0.88      7991


Accuracy Score:  0.8559629583281191
F1 Score:  0.4426150121065375
Cross Validation Error:  0.3529068165446525
```





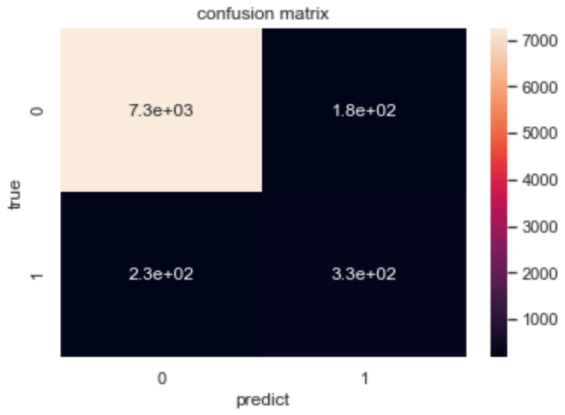
1.5.3 SVM

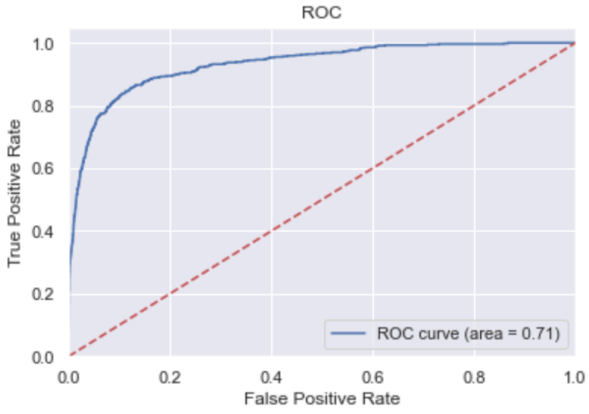
```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      7432
           1       0.65      0.59      0.62       559

    accuracy                           0.95      7991
   macro avg       0.81      0.78      0.80      7991
weighted avg       0.95      0.95      0.95      7991


Accuracy Score:  0.948942560380428
F1 Score:  0.6194029850746268
Cross Validation Error:  0.21797383375584262
```





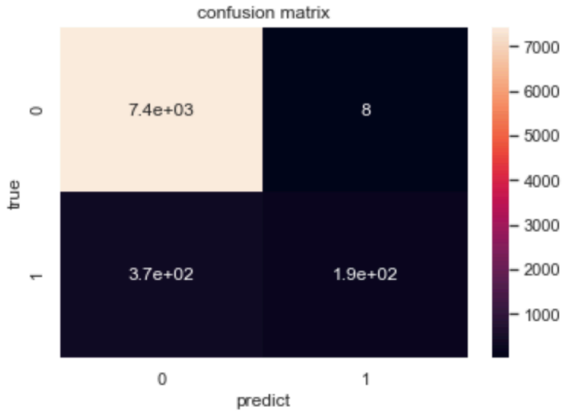
1.5.4 Random Forest

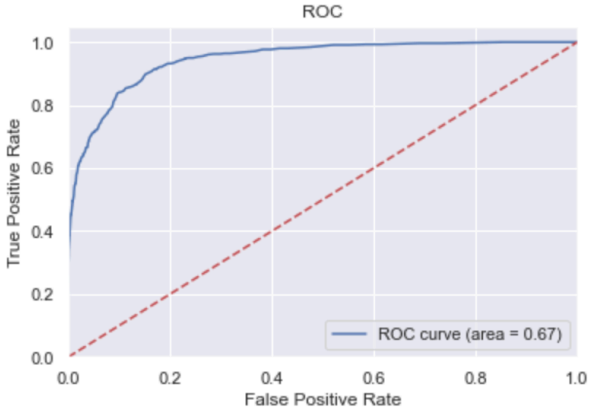
```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.98      7432
           1       0.96      0.35      0.51       559

    accuracy                           0.95      7991
   macro avg       0.96      0.67      0.74      7991
weighted avg       0.95      0.95      0.94      7991


Accuracy Score:  0.9531973470153923
F1 Score:  0.5078947368421053
Cross Validation Error:  0.2188078243953031
```





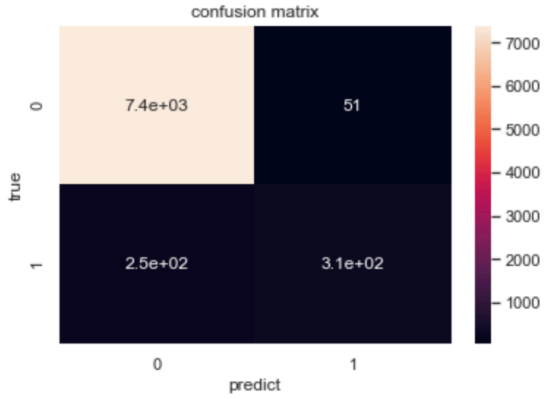
1.5.5 XGBoost

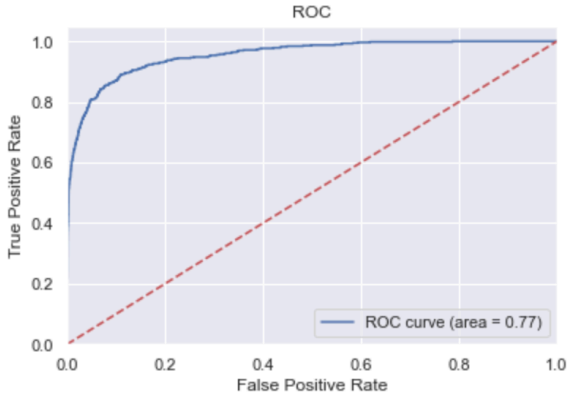
```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      7432
           1       0.86      0.56      0.68       559

    accuracy                           0.96      7991
   macro avg       0.91      0.77      0.83      7991
weighted avg       0.96      0.96      0.96      7991


Accuracy Score:  0.9625829057689901
F1 Score:  0.6753528773072747
Cross Validation Error:  0.19468668129994463
```





1.6 Report which classifier performs better than the others and describe the methods you use to reach this conclusion

	Model_ID	F1Score	Accuracy
0	Logistic Regression	0.618538	0.946440
1	Naive Bayes	0.428041	0.845827
2	SVM Classifier	0.612963	0.947691
3	Random Forest Classifier	0.514286	0.953197
4	XGBoost	0.656555	0.960330

In the unified use of Word2Vec for word embedding, models such as Logistic regression, SVM, Random Forest and XGBoost were evaluated respectively. Combining all evaluation indicators, focusing on F1 scores, we believe that the best performing model is XGBoost, with an F1 score of 0.656.