A Report on Twitter Sentimental Analysis

Priyanka Budavi
p_budavi@uncg.edu

***Abstract*— Sentiment analysis is a type of natural language processing for tracking the mood of the public about a particular product or topic. It deals with identifying and classifying opinions or sentiments expressed in source text. These days, social media is generating a vast amount of sentiment rich data in the form of tweets, status updates, blog posts etc. Thus, the data generated is very useful in knowing the opinion of the crowd. Also, data extracted is extremely valuable to the companies who sell products and get the information from the tweets regarding their product which in turn help them to improve their business or gain profit. Twitter sentiment analysis is difficult compared to general sentiment analysis due to the presence of slang words and misspellings and the maximum limit of characters that are allowed in Twitter which is 140. The objective of this project is to detect hate speech in tweets. For the sake of simplicity, we say a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from the tweets.**

I. INTRODUCTION

In the era of the popularity of Social Networking Service (SNS), people became increasingly inseparable from mobile phones and computers. People want to get information and real-time updates from social media, and they want to know how many Internet citizens have comments and opinions on many dynamic news. The interaction among users on social networking platforms is usually positive, advisory, motivating and influential. However, sometimes people will also reveal objectionable content, such as hate speech, abusive and bullying or discriminatory words. In recent years, social networks (and especially Twitter) have been used to spread hate messages. Hate speech refers to a kind of speech that denigrates a person or multiple persons based on their membership to a group, usually defined by race, ethnicity, sexual orientation, gender identity, disability, religion, political affiliation, or views.

II. METHODOLOGY

A. *Data Cleaning*

Data mining approach is followed for the process. A data mining approach typically includes phases such as data understanding, data preparation, modeling, and evaluation. The data set obtained from twitter contains unwanted characters which needs to be cleaned and analysed before data modelling process. It is said that the pre-processing of the text data is an essential step as it makes the raw text ready for mining, i.e., it becomes easier to extract information from the text and apply machine learning algorithms to it. If we skip this step then there is a higher chance that we will be working with noisy and inconsistent data. The objective of this step is to clean noise those are less relevant to find the sentiment of tweets such as punctuation, special characters, numbers and masked @user which don't carry much weightage in context to the text. The Figure 2 and 3 represent the data-set with label zero and label one i.e racist or sexist tweets being 1 and non racist and sexist being 0. The Figure 3 represents the length of the tweets column from the dataset for both training and the testing. We calculate this by using the str.len() function to the column tweet in the dataset. I collected the dataset from **https://www.kaggle.com**

```
In [5]: train[train['label'] == 1].head(10)
Out[5]:
```

	id	label	tweet
13	14	1	@user #cnn calls #michigan middle school 'build the wall' chant '' #tcot
14	15	1	no comment! in #australia #opkillingbay #seashepherd #helpcovedolphins #thecove #helpcovedolphins
17	18	1	retweet if you agree!
23	24	1	@user @user lumpy says i am a . prove it lumpy.
34	35	1	it's unbelievable that in the 21st century we'd need something like this. again. #neverump #xenophobia
56	57	1	@user lets fight against #love #peace
68	69	1	ð□□©the white establishment can't have blk folx running around loving themselves and promoting our greatness
77	78	1	@user hey, white people: you can call people 'white' by @user #race #identity #medâ□¦
82	83	1	how the #altright uses & insecurity to lure men into #whitesupremacy
111	112	1	@user i'm not interested in a #linguistics that doesn't address #race & . racism is about #power. #raciolinguistics bringsâ□¦

Fig. 1. Data-set with label 1

```
train[train['label'] == 0].head(10)
```

	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 !

Fig. 2. Data-set with label 0

*1) **Removing @user from the tweets**:* Several tweets will have the user information as the comment belongs to which user. This information does not carry any weight-age so as to determine the positivity and negativity of a comment. Hence these words have to be eliminated from the data. Figure 4 shows the words removed from the tweet column.

```
In [8]: length_train = train['tweet'].str.len()
        length_test = test['tweet'].str.len()
        plt.hist(length_train, bins=20, label="train_tweets" )
        plt.hist(length_test, bins=20, label="test_tweets" , color = "yellow")
        plt.legend()
        plt.show()
```

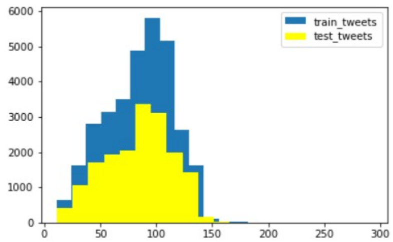


Fig. 3. Length of the Data-set

3. Removing Short Words

```
In [18]: combi['tidy_tweet'] = combi['tidy_tweet'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))

In [20]: combi.head(10)
```

Out[20]:

	id	label	tweet	tidy_tweet
0	1	0.0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run	when father dysfunctional selfish drags kids into dysfunction #run
1	2	0.0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked	thanks #lyft credit cause they offer wheelchair vans #disapointed #getthanked
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in urð□□±!!! ð□□□ð□□□ð□□□ð□□□ ð□□¦ð□□¦ð□□¦	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguide society #motivation
5	6	0.0	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo	huge fare talking before they leave chaos disputes when they there #allshowandnogo
6	7	0.0	@user camping tomorrow @user @user @user @user @user @user @user dannyâ□¦	camping tomorrow danny
7	8	0.0	the next school year is the year for exams.ð□□¯ can't think about that ð□□ #school #exams #hate #imagine #actorslife #revolutionschool #girl	next school year year exams think about that #school #exams #hate #imagine #actorslife #revolutionschool #girl
8	9	0.0	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers â□¦	love land #allin #cavs #champions #cleveland #clevelandcavaliers
9	10	0.0	@user @user welcome here ! i'm it's so #gr8 !	welcome here

Fig. 4. Removed short words in the tweet column.

2) ***Removing punctuation , special characters and numbers.:*** The special characters , numbers and punctuation are removed as they do not give meaningful information. Refer Figure 5 in the tweet column to see the cleaned column.

3) ***Text Normalization:*** Once the unwanted words are removed, its necessary to separate each word from a tweet and this is called tokenization and each word is a token. Thus, tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.This step is important because we want to understand what words have been used more frequently. That way we can visualize the positive words and also the negative words using word-cloud. Also, one of the most popular stemming algorithms is the Porter stemmer, where we can stem the words to the root words. Figure 6 , 7 show the normalization and stemming.

4) ***Word Cloud :*** Word Clouds also known as text cloud are used to highlight the words that are often used in a sentence. The bigger and bolder the word appears, the more often it's mentioned within a given text and the more important it is. Creating these word clouds helps us understand what words have been used more and what are they. For the project I segregated the words with label 0 and label 1 to understand the positive and negative words in the data-set. In the Fig 8 , 9 and 10 we observe the word clouds generated with the whole dataset , with label 1 and label 0.

2. Removing special characters , numbers and punctuation

```
In [19]: combi['tidy_tweet'] = combi['tidy_tweet'].str.replace("[^a-zA-Z#]", " " , regex=True)
         combi.head(10)
```

Out[19]:

	id	label	tweet	tidy_tweet
0	1	0.0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction #run
1	2	0.0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked	thanks for #lyft credit i can t use cause they don t offer wheelchair vans in pdx #disapointed #getthanked
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in urð□□±!!! ð□□□ð□□□ð□□□ð□□□ ð□□¦ð□□¦ð□□¦	#model i love u take with u all the time in ur
4	5	0.0	factsguide: society now #motivation	factsguide society now #motivation
5	6	0.0	[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo	huge fan fare and big talking before they leave chaos and pay disputes when they get there #allshowandnogo
6	7	0.0	@user camping tomorrow @user @user @user @user @user @user @user dannyâ□¦	camping tomorrow danny
7	8	0.0	the next school year is the year for exams.ð□□¯ can't think about that ð□□ #school #exams #hate #imagine #actorslife #revolutionschool #girl	the next school year is the year for exams can t think about that #school #exams #hate #imagine #actorslife #revolutionschool #girl
8	9	0.0	we won!!! love the land!!! #allin #cavs #champions #cleveland #clevelandcavaliers â□¦	we won love the land #allin #cavs #champions #cleveland #clevelandcavaliers
9	10	0.0	@user @user welcome here ! i'm it's so #gr8 !	welcome here i m it s so #gr

Fig. 5. Removed punctuation , numbers and special characters.

```
In [52]: tokenized_tweet = combi['tidy_tweet'].apply(lambda x: x.split())

In [53]: tokenized_tweet.head()

Out[53]: 0              [when, father, dysfunctional, selfish, drags, kids, into, dysfunction, #run]
         1    [thanks, #lyft, credit, cause, they, offer, wheelchair, vans, #disapointed, #getthanked]
         2                                                                   [bihday, your, majesty]
         3                                                         [#model, love, take, with, time]
         4                                                       [factsguide, society, #motivation]
         Name: tidy_tweet, dtype: object
```

Fig. 6. Tokenized and Normalized words in the tweet column

B. Bar Graph of the cleaned data

Once the data is cleaned its necessary to observe the pattern of words in the bar graph. For this I wrote a function which reads hashtags from the tweets and stores in an array. Later, I try to plot a graph with most often used hashtags to see how many times a hashtag is being used in a tweet. The bar plots for both label 0 and label 1 is plotted to understand the count of hashtags. This can be identified by ' # ' in the tweet column of the data-set. In the code the findall() function tries to look for the pattern in the column. The read word is stored in an array and later based on label we are displaying the hastags that are most repeated. Figure 11 , 12 and 13 show the bargraph plot of words and code to implement that.

C. Extracting tweets from a cleaned data-set

1) ***Bag of Words:*** To analyze the pre-processed data its necessary to convert the tweets into features. Depending on the number of usage of a word we can construct features using techniques like bag-of-words or TF-IDF. I am using Bag-of words as my technique to create features.

In this technique, a text is represented into numerical features. The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.

PorterStemmer() is a function to normalize the tweets.

```
In [24]: from nltk.stem.porter import *
         stemmer = PorterStemmer()

         tokenized_tweet = tokenized_tweet.apply(lambda x: [stemmer.stem(i) for i in x])
```

```
In [28]: for i in range(len(tokenized_tweet)):
             tokenized_tweet[i] = ' '.join(tokenized_tweet[i])

         combi['tidy_tweet'] = tokenized_tweet
```

Fig. 7. Stemming of the words

```
negative_words = ' '.join([text for text in combi['tidy_tweet'][combi['label'] == 1]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(negative_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

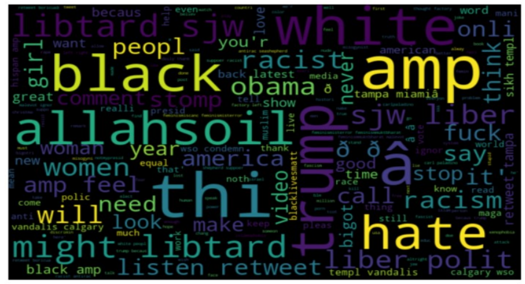


Fig. 9. Word Cloud with label 1

```
all_words = ' '.join([text for text in combi['tidy_tweet']])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(all_words)

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

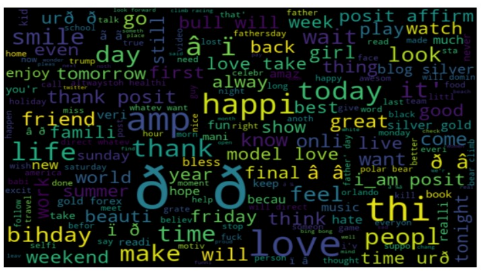


Fig. 8. Word Cloud of the dataset

```
normal_words =' '.join([text for text in combi['tidy_tweet'][combi['label'] == 0]])

wordcloud = WordCloud(background_color='black' , width=800, height=500, random_state=21, max_font_size=110).generate(normal_word
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

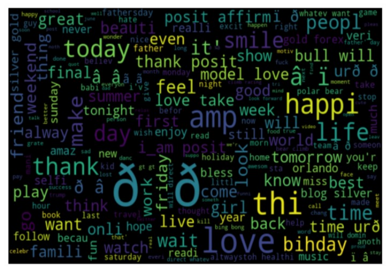


Fig. 10. Word Cloud with label 0

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:
1. A vocabulary of known words.
2. A measure of the presence of known words.

It is called a “bag” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

D. *Training the model*

*1) **Logistic Regression :*** After cleaning the data-set , now comes the task for training the model. Training is a process of optimizing the model parameters using a set of labeled data sets. For this we use python libraries to train the model. Figure 13 shows the code to fit the model using logistic regression. I used the sklearn.linear model. I used the logistic model because the classification is discrete i.e the comments are going to be either positive or negative.Firstly, I import the Logistic Regression module and create a Logistic Regression classifier object using Logistic Regression() function. Then, fit the model on the train set using fit() and perform prediction on the test set using predict(). The predicted values obtained are between 0 to 1. Thus I calculate the average which comes around 0.3 so any value greater than 0.3 is labeled or classified as label 1 and lesser than the mean value is classified as label 0. Figure 14 shows the code for logistic regression.

*2) **Support Vector Machine :*** Support vector machines (SVMs) are a set of supervised learning methods used for classification and regression. Support Vector Machine is one of the classical machine learning techniques that can help solve big data classification problems. Especially, it can help the multi-domain applications in a big data environment. However, the support vector machine is mathematically complex and computationally expensive. Figure 15 shows the code for training the data using SVM.

III. RESULTS AND CONCLUSION

A. *Results*

To calculate and compare the measure I used F1 score because the data is imbalanced and F1 score is the best measure when it comes to dealing with such datasets. The Logistic Regregression model performed better than Support Vector Machine. Also, the technique used is Bag-of-words. If we change the technique and use Word2Vec there is a possibility of getting a better score. Figure 16 and 17 show the Predicted labels for the dataset. Now, the question arises when to use which model. Depending on the number of observations and features that we have, one can choose to use either logistic regression or support vector machine. It

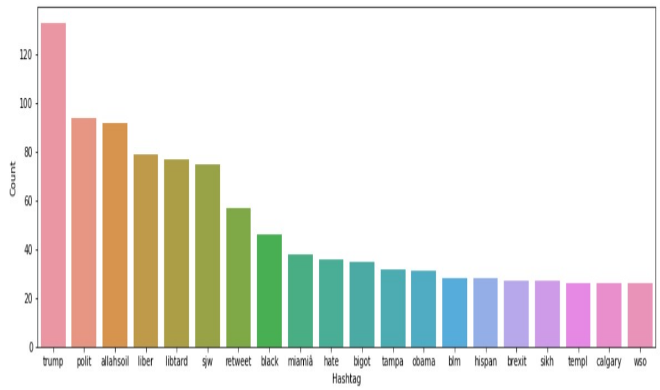


Fig. 11. Hashtag with label 1

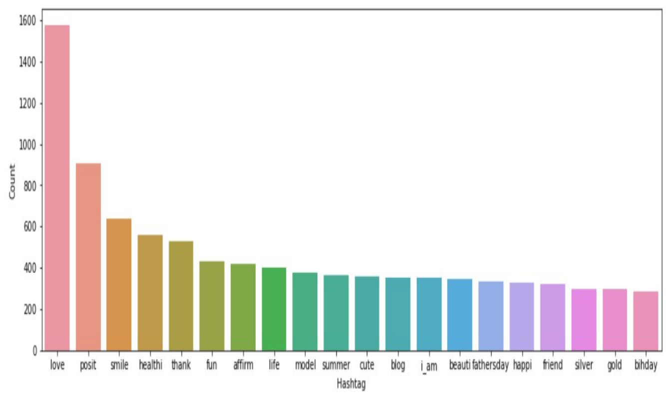


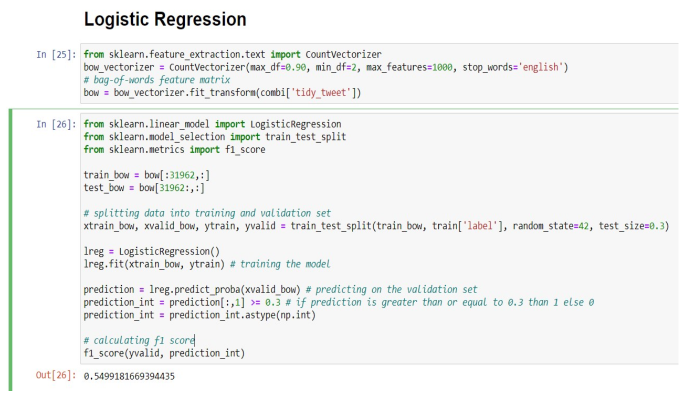
Fig. 12. HashTag with label 0

Extracting the HashTags with label zero and label 1

```
In [22]:
# extracting hashtags from non racist/sexist tweets

HT_regular = hashtag_extract(combi['tidy_tweet'][combi['label'] == 0])

# extracting hashtags from racist/sexist tweets
HT_negative = hashtag_extract(combi['tidy_tweet'][combi['label'] == 1])

# unnesting list
HT_regular = sum(HT_regular,[])
HT_negative = sum(HT_negative,[])
```

```
In [23]:
a = nltk.FreqDist(HT_regular)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```

Fig. 13. Code to extract hastags from the tweet column



Logistic Regression

```
In [25]: from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
# bag-of-words feature matrix
bow = bow_vectorizer.fit_transform(combi['tidy_tweet'])
```

```
In [26]: from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

train_bow = bow[:31962,:]
test_bow = bow[31962:,:]

# splitting data into training and validation set
xtrain_bow, xvalid_bow, ytrain, yvalid = train_test_split(train_bow, train['label'], random_state=42, test_size=0.3)

lreg = LogisticRegression()
lreg.fit(xtrain_bow, ytrain) # training the model

prediction = lreg.predict_proba(xvalid_bow) # predicting on the validation set
prediction_int = prediction[:,1] >= 0.3 # if prediction is greater than or equal to 0.3 than 1 else 0
prediction_int = prediction_int.astype(np.int)

# calculating f1 score
f1_score(yvalid, prediction_int)

Out[26]: 0.5499181669394435
```

Fig. 14. Code to train the model using Logistic Regression

is recommended that if the number of features are greater than the observations, then use Logistic Regression and if the number of observations are more than the features, SVM is used. Generally, it is advisable to first try to use logistic regression to see how the model does, if it fails then one can try using SVM without a kernel (is otherwise known as SVM with a linear kernel). Logistic regression and SVM with a linear kernel have similar performance but depending on the features, one may be more efficient than the other. Also, the F1 score may vary because it depends on how clean the data is.

B. *Conclusion*

The Machine Learning concepts play a vital role in these days in order to attract people towards making profits. There are number of applications which have incorporated machine learning concepts to make profits. The best examples are Amazon , Facebook etc.,how they are attracting the customers by displaying the ad's that might interest the user. From the above experiment, I have learned the importance of looking in to statistical details of data set and finding hidden patterns. It is also important to discard some features which are not helpful in prediction process. The process followed in this paper helped me to connect with the concepts taught in the course. The way data is extracted, statistical details are drawn, model training and testing, quantitative evaluation of predictions to find out the error in predictions and trying the achieve the best accuracy. One potential problem with my experiment is that the sizes of the classes are not equal. The problem with unequal classes is that the classifier tries to increase the overall accuracy of the system by increasing the accuracy of the majority class, even if that comes at the cost of decrease in accuracy of the minority classes. In future, I would like to work with balanced data-set and optimize my existing work.

References

[1] A.Pak and P. Paroubek. „Twitter as a Corpus for Sentiment Analysis and Opinion Mining". In Proceedings of the Seventh Conference on International Language Resources and Evaluation, 2010, pp.1320-1326 Poster Volume,pp. 36-44.

[2] R. Parikh and M. Movassate, "Sentiment Analysis of User- GeneratedTwitter Updates using Various Classi_cation Techniques",CS224N Final Report, 2009

[3] Go, R. Bhayani, L.Huang. "Twitter Sentiment ClassificationUsing Distant Supervision". Stanford University, Technical Paper,2009

Support Vector Machine

In [29]:
```python
from sklearn import svm
```

Bag of words

In [30]:
```python
svc = svm.SVC(kernel='linear', C=1, probability=True).fit(xtrain_bow, ytrain)

prediction = svc.predict_proba(xvalid_bow)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)
f1_score(yvalid, prediction_int)
```

Out[30]: 0.5255474452554744

In [31]:
```python
test_pred = svc.predict_proba(test_bow)
test_pred_int = test_pred[:,1] >= 0.3
test_pred_int = test_pred_int.astype(np.int)
test['label'] = test_pred_int
submission = test[['id','label']]
submission.to_csv('sub_svc_bow.csv', index=False)
```

Fig. 15. Code to train the model using SVM Model

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
1
0
0
0
0
0
0
0
0
0
0
0
1
0
0
0
0
0
0
```

Fig. 16. Predicted values for Logistic Regression Model

```
id,label
31963,0
31964,0
31965,0
31966,0
31967,0
31968,0
31969,0
31970,0
31971,0
31972,0
31973,0
31974,0
31975,0
31976,0
31977,0
31978,0
31979,0
31980,0
31981,0
31982,1
31983,0
31984,0
31985,0
31986,0
31987,0
31988,0
31989,0
31990,0
31991,0
31992,0
31993,1
31994,0
31995,0
31996,1
31997,0
31998,0
31999,0
```

Fig. 17. Predicted values for SVM Model