COMPUTER SCIENCE DEPARTMENT

UNIVERSITY COLLEGE LONDON

Sentiment Analysis with Twitter Data

COMPGI15 Group Project

NAME: YUE WANG (15090180)

XIZHE JIANG (15005515)

SHUO WANG (15033967)

Github:

https://github.com/imink/UCL_COMPIG15_Project.git

DATE: APIRL, 2016

CONTENTS

1. INTRODUCTION

Sentiment analysis aims to judge whether the emotional tend expressed in a text is positive or negative. Generally speaking, sentiment analysis is trying to determine the attitude of a speaker or a writer on a topic or the overall contextual polarity of a document. The basic task is classifying the polarity of the given text. In this project, it focused on the Tweeter, a popular microblog. The various models are built to classify the tweets into positive or negative sentiment. Before using the texts, there are several ways to pre-process the data. Moreover, then the feature extraction converts the twitter texts to a vector. There are three methods to obtain the feature: TI-IDF, TF, and FP. Moreover, the five supervised learning classifiers are implemented, including Naïve Bayes, Logistic regression, SVM, DecisionTrees, and KNN. Finally, the evaluation is used for these five models, and the SVM achieved the best performance.

2. METHODOLOGY

The dataset is pre-processed to clear the noise information from the twitters, and it makes the Twitter becoming more meaning, such as removing hashtag and stemming. After that, there are various fetching feature methods and machine learning algorithm for the classifiers training. Finally, the cross-validation is used to test the accuracy of the above trained models.

2.1 Dataset

In this project, the dataset is found from the internet[1], which is based on data from the following two sources, University of Michigan Sentiment Analysis competition on Kaggle and Twitter Sentiment Corpus by Niek Sanders. The original Twitter Sentiment Analysis Dataset contains 1578627 classified tweets. Each tweet has been labeled by 1 or 0, denoting positive or negative. It is too large a dataset to be used. Thus, two datasets are loaded separately. The first on is top 5000 sequent Twitter, which has 1445 negative tweets and 3554 positive tweets. Furthermore, after the tweets ranked by ascending order, the 5000 tweets are selected randomly to be used in this project as the 5000 random data. It is declared that 10% of the sentiment classification can be debated, and the maximum relative accuracy can be achieved is 90%. However, there are lots of usable characters in the tweets need to be processed, which will be discussed in the later section. An example of the data that is used after processing is illustrated as bellow in Table 1:

Index	Label	Source	Content
4432	0	Sentiment140	Poor lady. I wish I could help.

Table 1: The raw data structure

The main challenges of processing tweets emerge because of the size restriction of 140 characters. This limitation makes people use shortcuts, omit prepositions short links and "@" to refer to another tweet. People also use hashtags in the tweet and special characters which should be pre-processed before training and evaluating. Related disturbances will be pre-processed and leave the pure content with readable characters. The detail about how to random choose data from the original can be described as below:

```
rand_var = random.sample(range(1, 150000), 5000)
rand_var.sort()
print rand_var
j = 0
with open('5000_train.csv', 'aw+') as outputfile, open('data.csv', 'r') as inputfile:
        for i, line in enumerate(inputfile):
            if i == rand_var[j]:
                # print i, rand_var[j]
                outputfile.write(line)
                j   = j + 1
            if j >= 5000:
                break
```

2.2 Tools

To apply the data processing and the various classifier, the Python is selected as the main coding language. The reason using this method is that there are various external libraries to support the Twitter sentiment analysis. The main selected libraries are the scikit-learn and NLTK (natural language toolkit), which includes the kinds of methods for machine learning and natural language processing. The scikit-learn provides a set of training model with a friendly interface, including a range of supervised and unsupervised algorithms [2]. Also, the NLTk is an efficient platform for processing the human language data, which has the following facilities: classification, tokenization, stemming[3]. Bothe of them are supported with other libraries, like Matplotlib, Numpy, and Scipy.

2.3 Pre-processing

Raw data from the original dataset are useless to data training. It takes a few steps to normalize the text. To achieve valuable terms from the original data, the data needs to be tokenized and vectored the raw data and remove irrelevant characters. It involves Pandas firstly to extract columns in the raw data and formalize them into vectors. However, there are lots of unusable characters need to be pre-processed, such as "@", links, digits hashtags, etc. Furthermore, the regular expression is used to match and filter the characters to remove. The useless characters that are handled can be concluded as below.

Hashtags ("#"), short and long links(e.g. "http|https"), Lemmatization (find stems of the original word), Punctuations, repeating characters, special characters like "&quot," "@" characters, etc.

Although some Punctuations are important from classification, they are not considered to the contribution of the punctuations in this project. The scikit-learn framework can convert a collection of text documents to a matrix of token counts from a CSV file. The CountVectorizer can also be used to accept stop-word arguments. In that case, a built-in stop-word list for English will be used to remove the punctuations from the resulting tokens of CountVectorizer. However, this cannot fully deal with the unusable characters. Furthermore, the links are removed, hashtags and "@" as well as multiplications which are the repeat characters.

It can also use regular expressions to match the repeat expression by "(.)\1{1,}" and "(http|https|ftp)://[a-zA-Z0-9\\./]+" to match all the links in the tweets. "@" character can be matched by "@(\w+)" and "#(\w+)" to match hashtags. In addition, the repeat characters are processed in a word, such as "Yeaaaaaaaaaah". To obtain a precision TF-IDF score, this word has to be recovered to "Yeah" instead. For each word, the unusable characters are removed in that particular word.

2.4 Stemming & Lemmatization

The stemming algorithms involve the affix removing, which is the basic concepts, applying a set of the cutting rules to each word. The normally known prefixes and suffixes are removed, and the words will be recognized as one word [4]. For example, "books" is changed to "book". The first stemming was created in 1968, the words which have an ending with matching the condition are retrieved, and the suffixes are cut. Recently, there are three different

stemmers used widely: Porter, Snowball, and Lancaster. The Porter stemmer is used commonly, which can offer a balance between speed and accuracy. Furthermore, the snowball is an improved version of the porter in the computation and various language. Meanwhile, the Lancaster Stemmer is another choice, which is more aggressive than the other methods to reduce the set of words hugely [5]. After applying this approach, the representations of the part of words are not readable, but it is a fast algorithm.

The Lemmatization also is a way to normalizing words. The main difference is that the lemmatization transfers the words to their original representation, rather than cut the different section. The NLTK library involves the WordNet Database to lookup lemmas. In fact, the performance of this method is not satisfied, and the dataset might need to be improved.

2.5 Features

To make that the machine could read the text, the features are important to transfer the text to a vector. There are three ways to convert. The both TF and TF-IDF indicating the frequency of the feature in out dataset.TF measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length to achieve the normalization

TF(t) = (n(term) in a document) / (Total (terms) in the document)

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important.

IDF(t) = log_e (Total (documents) / N (documents with) term t in it).

Moreover, **FP (Feature presence)** method is selected as the feature, it a binary value, indicating whether the feature exists in the text. Only the word exist in the vocabulary will be set to 1, and others will be set to 0. The vocabulary set is the set of all the words in the corpus. In addition, the threshold is set for FP, for example, if it is configured the threshold to 5, then only the input value is more than five will set the value to be 1. However, after evaluation, it describes that the TF-IDF provided the highest precision among TF, TF-IDF, PF as features.

3. CLASSIFIER

3.1 Naïve Bayes

The Naïve Bayes classifier is a probabilistic classifier, which is based on the Bayes theorem. In the basic Bayes theorem, the result is the conditional probability of finding the event A given then event B being true.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using simple transforming converts it to the following equation:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

In this project, the given label y and their feature vector x_n that is transferred from the twitter content place the variables A and B. Since the $P(x_1, \dots, x_n)$ is the same with the given inputs, it will be ignored. Therefore, it is shown in the below formulary:

$$P(y|x_1, \dots, x_n) = P(y)P(x_1, \dots, x_n|y)$$

Also, with the naïve independence theory, the equation becomes that:

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Thus, in the predicting processes, the maximum value of the above equation is found, regard as the predicted label $\hat{y}$.

$$\hat{y} = \arg\max P(y) \prod_{i=1}^{n} P(x_i|y)$$

3.2 Logistic regression

The logistic regression also is a classification method, which is also called maximum entropy classification. The output result is the probability from 0 to 1. The prediction function **h** is built with the input **x** the weight **w**. At the same time, the **h** is loaded to the sigmoid function which converts the value to (0, 1),

the sigmoid equation shown below and plot in Figure 1:

$$sigmoid(h) = \frac{1}{1 + e^{-h}}$$

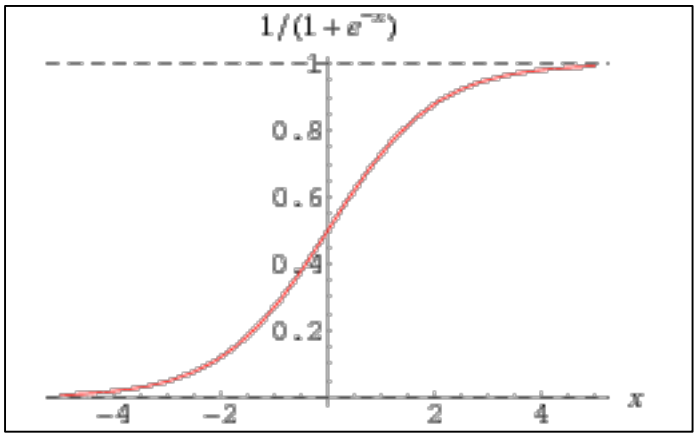

Figure 1: The Sigmoid function [6]

Furthermore, the cost function J is defined as the maximum squared error method with prediction h and the label y. After that, the purpose of the logistic regression is that using the gradient descent obtain the weight **w** which produces the minimum value of the cost function. This weight is used to predict with the L2 regularization.

3.3 Support Vector Machines

SVM is a popular supervised learning model with associated learning algorithms that analyse data and recognise patterns for classification and regression. The SVM model trained via the labelled data then builds a non-probabilistic linear classifier. The points from the training set are divided by a hyperplane that has the largest margin. The decision function is shown as this hyperplane. Moreover, the larger the margin is, the lower that generalisation error of the classifier. The advantages of this method are effective in a high dimension space and including the customised kernel function. The kernel function is a significant key in the SVMs, and in this project, the "rbf" kernel is selected. Furthermore, Figure 2 shows a simple example of SVMs model.

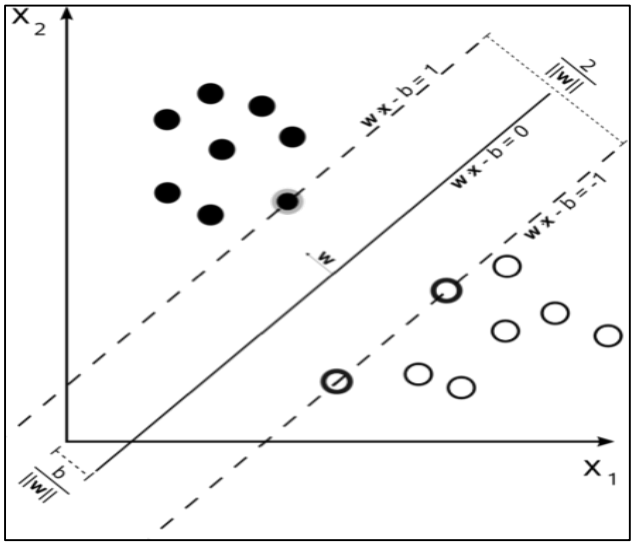

Figure 2: The SVMs example

3.4 Nearest Neighbours

The nearest neighbours method is a simple supervised learning way to classifying the data with the label. In this project, each Twitter in the training data is processed as a point in space. When a new point (unlabelled Twitter) is read to require the prediction, the k nearest neighbours of this new point in space will be found. For example, in Figure 3, if the k equals 3, there are 2 neighbours that belong to the class B. It means that this new point is predicted to the class B. However, if the K is 6, the new point has four neighbours in class A, and it will be regard as a point in class A. Therefore, the K value is the most import fact in this algorithm to train the model neither under-fitting nor over-fitting.

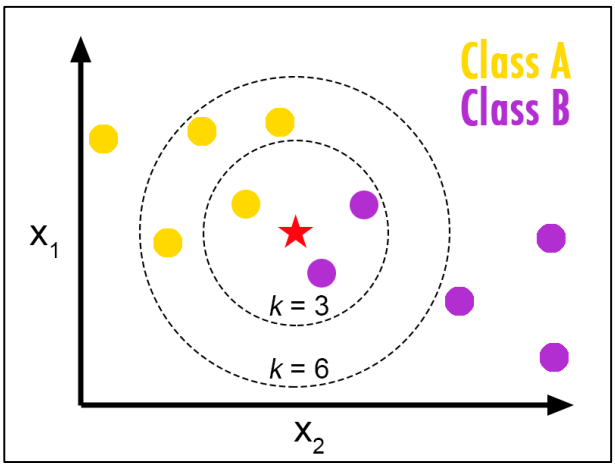

Figure 3: The KNN example [7]

3.5 Decision Tree

The decision trees are a supervised learning method, but it does not include the parameters. It predicts the value of the inputs via learning the decision rule from the training data. Also, for the decision tree classifier, it creates the finite and discrete set of values, and each leaf shows the class labels and the branches state the path that leads to the labels. An example is presented in Figure 4.

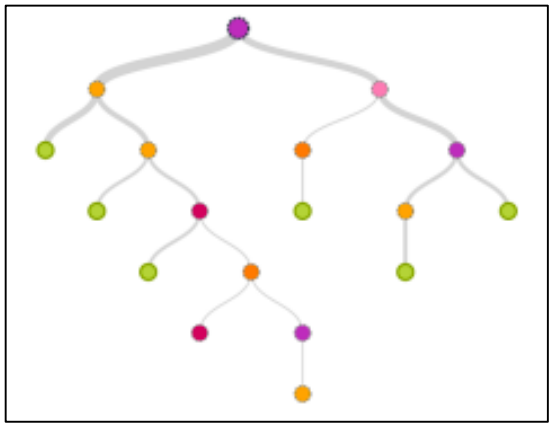

Figure 4: The Decision Trees example [8]

4. EVALUATION & DISCUSSION

4.1 Three evaluation approach

This section is aimed to describe the assessment of the proposed system mentioned before. A simple but effective method is demonstrating the feasibility of system by conduct the three evaluation mechanism: simple accuracy computation, k folding cross-validation, and f1 score. The simple accuracy tries to give a straightforward overall performance of the approached. This can be done by the following formula:

$$accuracy = \frac{\sum cortect\_pos + \sum correct\_neg}{total\ no\ of\ prediction} \times 100\%$$

The f1 score states a balanced and detailed evaluation of the performance, which describes both precisions and recalls for positive prediction and negative prediction respectively.

$$f1 = \frac{(precision \times recall)}{(precisoon + recall)} \times 100\%$$

A confusion metrics with the precision and recall should be illustrated as a key performance indicator of the supervised machine. In this section, the 10-fold cross-validation and get the average mean score.

The cross validation is the solution preventing over-fitting the data set, and a test set should be kept for final evaluation, but the validation set is no longer needed when doing CV. Therefore, the multi-fold cross-validation is applied, which splits the training set into k smaller sets, for each of the k folds, the model is trained by k -1 of the folding data, then the resulting model is validated on the remaining part of the data. The mean average of k folding cross-validation can measure the overall performance.

4.2 Performance comparison

From Figure 5 and Table 2, it can be found obviously that our SVM classifier achieves the highest score in our comparison for the same data set.

That might due to SVM fully considers the feature dependencies, which in a Twitter, the overall sentiment is dependent on a series of words not a single mutual word, for example, "Nice day today. I wanna get out of this place." means positive, however by only considering the latter part "get out of this place" might not distinguish its exact sentiment to be positive.

SVM does consider the inner relationship of the features, as s they provide a simple bridge from linearity to non-linearity for an algorithm that can be expressed regarding dot products, i.e. numbers, text, and graphics. They provide the operation in high-dimensional, implicit features space without computing the coordinates of the data in that space, but rather compute the inner products between the images of all pairs of information in the feature space. This operation is cheaper than the explicit computation of coordinates.

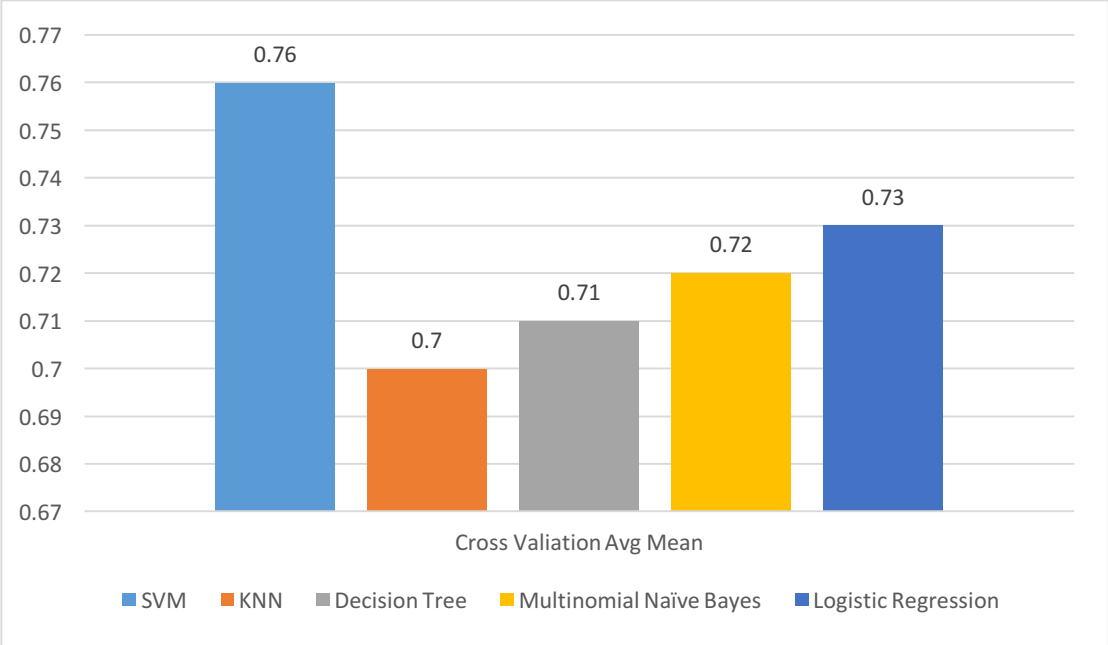

Figure 5: The score of the various classifiers

Classifier	SVM	KNN	Decision Tree	Multinomial Naïve Bayes	Logistic Regression
Simple Accuracy	0.75	0.72	0.69	0.74	0.75
Average 10-folds	0.76	0.70	0.71	0.72	0.73

Table 2: The score of the various classifiers

4.3 SVM analysis

In this section, the aim is to investigate the capability and performance of SVM classifier that outperforms others the last evaluation. The test is completed with two different datasets: 1) 5000_seq.csv, which the number of negatively labeled Twitter is more than positive ones; 2) 5000_random.csv, which the number of negatively labeled Twitters is less than positive ones.

Now the outcome is surprisingly different. As the previous dataset (more negative labeled Twitters), the trained SVM achieves 0.73 f1 scores and later dataset achieves 0.70 f1 scores. As for the precision and recall, for the dataset1, the prediction to negatively labelled twits outperforms the positive one as a whole. Moreover, classifier performs poorly on negative ones in the dataset2. To some extent, our SVM classifier is much capable of prediction the negative

twits rather than positive ones.

	Precision	Recall	F1-score	Support
Positive	0.76	0.93	0.84	1145
Negative	0.69	0.35	0.47	505
Avg / Total	0.74	0.75	0.73	1650

Table 3: The score of the 5000_seq

	Precision	Recall	F1-score	Support
Positive	0.70	0.58	0.63	722
Negative	0.71	0.81	0.76	933
Avg / Total	0.71	0.71	0.70	1655

Table 4: The score of the 5000_random

4.4 SVM parameter turning

As SVM classifier is the best categorization engine in our evaluation, there are some optimizations can be done by parameter turning. Moreover, in this section, it emphasizes on the SVM kernel and the picking of the best estimator.

By default, C-support vector categorization function of sicikit-learn using rbf kernel, and another kernel like the linear kernel. Although linear kernel is more competitive handling linearly separate data and performs faster than the non-linear kernel, RBF kernel is proved to be the best for text classification problems [8].

The two parameter c and gamma of rbf kernel are hyperparameters that affect the classification performance of the system. Influentially the gamma parameter constrains how far the influence of single training example reaches. Low gamma value indicates high influence while high gamma value is meaning "close." The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. The c parameter trades off the misclassification of training example against the simplicity of the decision interface. A low c meaning the decision of interface smoother, while a high C aims at classifying all training examples correctly by give the model freedom to select more samples as support vectors.

A grid search method has been applied to compute the best estimator. The scikit-learn built-in GridSearchVC function is important and makes a pipeline implement the whole process.

```
class sklearn.grid_search.GridSearchCV(estimator, param_grid, scoring=None,
loss_func=None, score_func=None, fit_params=None, n_jobs=1, iid=True, refit=True,
cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score='raise')
```

Figure 6: The GridSearchCV method

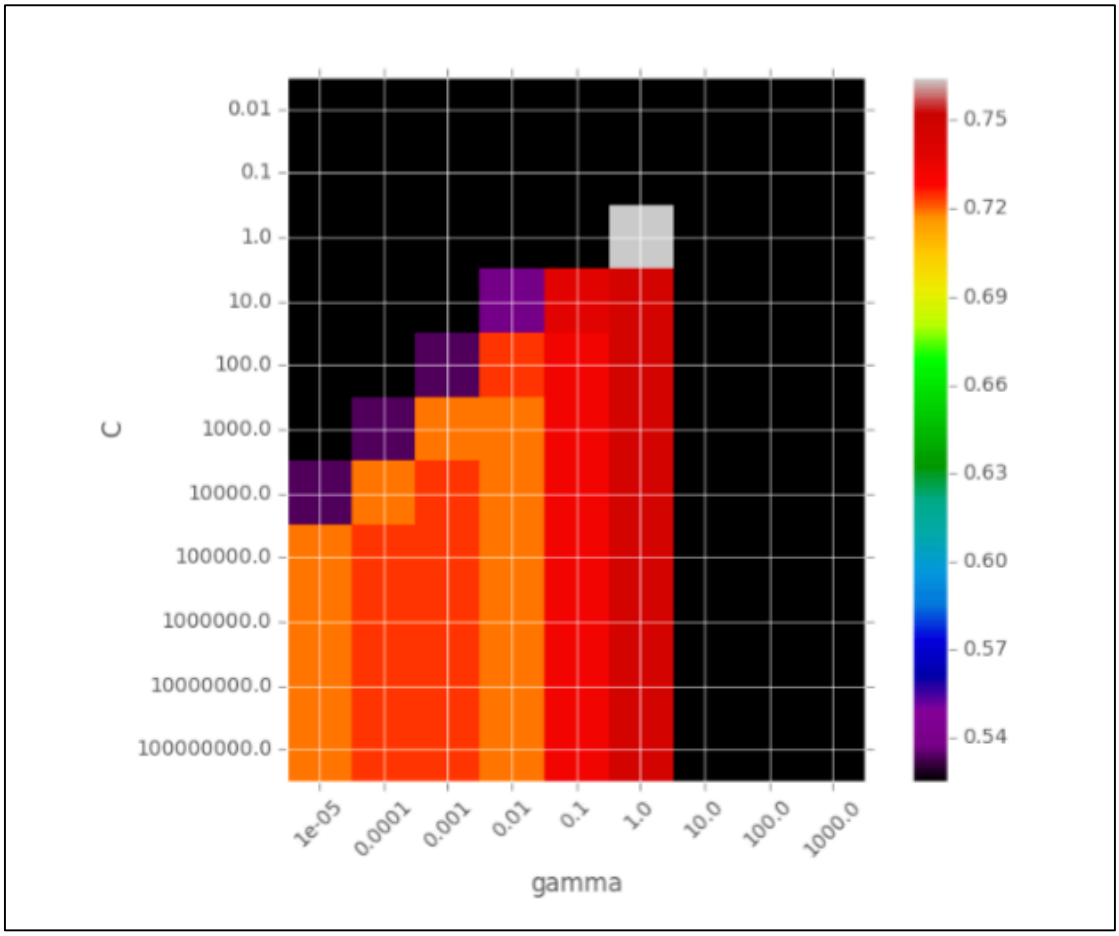

Figure 7: The score of various parameters for SVM

5. CONCLUSION

In general, this project achieved the different models for the Twitter sentiments analysis and classified the Twitter positive or negative. Firstly, the raw data is around 1500000, which needs to select part of them to train the models. There are two ways to select: the top 5000 sequential data and the 5000 random

datasets. After that, the data passes through the pre-processing function to remove and handle the unusable information. Before training step, the text data is converted to the vector via three different ways, TF-IDF, TF, FP. These vectors are loaded as the training data to the various classifiers, including Naïve Bayes, Logistic regression, SVM, DecisionTrees, and KNN. Finally, the k-fold cross-evaluation method is involved in testing the performance of each model and obtaining the best one. The project was completed with the cooperation of the external library like scikit-learn and NLTK.

In our experiment, we found that in all three feature extraction methods, TFIDF generate higher accuracy and performance. And in the classifiers mentioned above, SVM achieves better performance in prediction.

In the future, more data will be used to train the model, and the pre-processing step might be applied to achieve a more usable dataset. For example, the lemmatization whose dictionary will be improved to convert the words to their original version. In addition, an advanced model could be implemented in the real word to avoid the manual classification when the dataset is too huge.

6. REFERENCE LIST

Reference:
[1] Twitter Sentiment Analysis Training Corpus (Dataset)
Available:http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22
[Access: 13/04/2016]

[2] Brownlee, J. (2014) A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library
Available: http://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/
[Access: 13/04/2016]

[3] NLTK 3.0 documentation, Natural Language Toolkit
Available: http://www.nltk.org/
[Access: 13/04/2016]

[4] Smirnov, J. (2008). Overview of stemming algorithms, *Mechanical Translation*.
[Access: 10/04/2016]

[5] How Stemming and Lemmatization Works, Stemming and Lemmatization with Python NLTK
Available: http://text-processing.com/demo/stem/
[Access: 10/04/2016]

[6] Sigmoid Function, Wolfram MathWorld
Available: http://mathworld.wolfram.com/SigmoidFunction.html
[Access: 10/04/2016]

[7] DeWilde, B. k-Nearest Neighbor Algorithm, Classification of Hand-written Digits (3)
Available: http://bdewilde.github.io/blog/blogger/2012/10/26/classification-of-hand-written-digits-3/
[Access: 10/04/2016]

[8] A New Way to Visualize Decision Trees
Availabel:https://blog.bigml.com/2013/04/19/a-new-way-to-visualize-decision-trees
[Access: 10/04/2016]

[8] Falinouss, P., (2007). Stock trend prediction using news articles. Master's thesis, Lulea University of Technology, pp.1653–0187.
Available: http://epubl.ltu.se/1653-0187/2007/071/LTU-PB-EX-07071- SE.pdf
[Access: 14/04/2016]