SemEval-2019 Task 6: Identifying Offensive Language in Tweets Using Machine Learning

Merna Zakaria and **Rowan Tahseen** and **Nevine Said** and **Hassan Elsoudy**

Faculty of Engineering, Alexandria University
Department of Computer and Communication Engineering

Abstract

Twitter, as a social media is a very popular way of expressing opinions and interacting with other people in the online world. When taken in aggregation, tweets can provide a reflection of public sentiment towards events. In this paper, we provide a positive or negative sentiment on Twitter posts using a well-known machine learning method for text categorization. In addition, we use manually labeled (OFF/NOT) tweets to build a trained method to accomplish a task.

1 Introduction

Hate speech is an emotive concept which has no universally accepted definition in international human rights law.
Since there's not a single internationally accepted definition of hate speech. But in broad terms, hate speech is a communication that denigrates people on the basis of their membership to a particular group. This can include any form of expression, such as images, plays and songs as well as speech.
Some definitions even extend the concept of hate speech to include communications that foster a climate of prejudice and intolerance the thinking here is that these kinds of communications may fuel discrimination, hostility and violent attacks later on.
Unfortunately, many users engaging online,
either on social media, forums or blogs, will often have the risk of being targeted or harassed via abusive language.
Although hate speech is protected under the free speech provisions in the United States,
there are other countries, such as Canada, France, United Kingdom, and Germany, where there are laws prohibiting it as being promoting violence or social disorder.
Automated Detection is the operation of detecting matters with no or minimal human intervention in a controlled environment. Terms directly associated with automated learning are machine learning and supervised learning. Machine learning, a part of the artificial intelligence branch, is a scientific discipline dealing with the design and development of algorithms that allow machines to evolve behavior based on experience. Supervised learning is the machine learning task of deducting a function from labeled (supervised) training data.

2 Problem statement

Offensive language is pervasive in social media. Individuals frequently take advantage of the perceived anonymity of computer-mediated communication, using this to engage in behavior that many of them would not consider in real life. Online communities, social media platforms, and technology companies have been investing heavily in ways to cope with offensive language to prevent abusive behavior in social media.

One of the most effective strategies for tackling this problem is to use computational methods to identify offense, aggression, and hate speech in user-generated content (e.g. posts,

comments, microblogs, etc.). We have to classify whether the tweet is offensive or not.
Sub-task (A) : Offensive language identification

3 Classification Process

From the perspective of automatic text classification systems, classification task can be sequenced

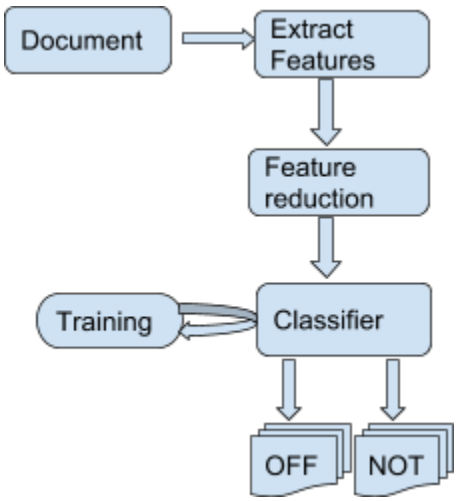

Figure 1: Description of the classification process

3.1 Preprocessing

Any piece of text which is not relevant to the context of the data and the end-output can be specified as the noise.
So we have to through out any unnecessary data to avoid overfitting.

3.2 Stop words (Noise)

In computing, stop words are words which are filtered out before or after processing of natural language data (text). Though "stop words" usually refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools even use such a list. Some tools specifically avoid removing these stop words to support phrase search. We decided to use our own pre-defined list which contains the most common English words mentioned here and we added the word "null" to our list remove it from data.

3.3 Removing non-Alphabetical words

After getting our free-noise list, we went through step 2, removing non-alphabetical words (e.g. words contain numeric value or special character) we used Python's built in function `isalpha()` to check whether the word contains non-alphabetical characters or not.

4 Processing

Text Analysis is a major application field for machine learning algorithms. However, the raw data, a sequence of symbols cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

```
a, about, after, all, also, an, another, any, and, are, as, and, at,
be, because, been, before, being, between, but, both, by, came, can, come,
could, did, do, each, even, for, from, further, furthermore, get, got, has, had,
he, have, her, here, him, himself, his, how, hi, however, i, if, in,
into, is, it, its, indeed, just, like, made, many, me, might, more, moreover,
most, much, must, my never, not, now of, on, only, other, our, out, or,
over, said, same, see, should, since, she, some, still, such, take, than,
that, the, their, them, then, there, these, therefore, they, this, those,
through, to, too, thus, under, up, was, way, we, well, were, what, when,
where, which, while, who, will, with, would, your, null
```

Figure 2: List of redundant words

In order to address this, *scikit-learn* provides utilities for the most common ways to extract numerical features from text content, namely:

- **tokenizing** strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.
- **counting** the occurrences of tokens in each document.
- **normalizing** and weighting with diminishing importance tokens that occur in the majority of samples / documents.

4.1 Extracting Features

We decided to use *TFIDF* (short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.) library, which is product of two statistics, *term frequency* and *inverse document frequency*

4.1.1 TF (Term frequency)

In the case of the term frequency *TF(t,d)*, we chose the simplest choice which to use the raw count of a term in a document, (i.e., the number of times that term t occurs in document d)

4.1.2 IDF (Inverse Document Frequency)

The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents.

$$\operatorname{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\operatorname{tf}(t,d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

4.1.3 TF-IDF (term frequency–inverse document frequency)

A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf–idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf–idf closer to 0

$$\operatorname{tfidf}(t, d, D) = \operatorname{tf}(t, d) \cdot \operatorname{idf}(t, D)$$

5. Classifiers

An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

5.1 SVM Classifier

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

5.1.1 Classification-Classes

This line not only separates the two classes but also stays as far away from the closest training instances as possible. You can think of an SVM classifier as fitting the widest possible street (represented by the parallel dashed lines) between the classes. This is called large margin classification.
SVC, NuSVC and LinearSVC are classes capable of performing multi-class classification on a dataset, in this problem we decided to use *LinearSVC* class, note that *LinearSVC* is similar to SVC with parameter `kernel='linear'`, it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.
The objective is to find a good balance between keeping the street as large as possible and limiting the margin violations.
We control this using C hyperParameter.

5.1.2 Regularization parameters

The Regularization parameter termed as C parameter in LinearSVC tells the SVM optimization how much we want to avoid misclassifying each training example. for large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassified more points.

5.1.3 Decision Boundaries

So after tuning parameters to choose the best values avoid Overfitting as we have to choose the best C for classifier:

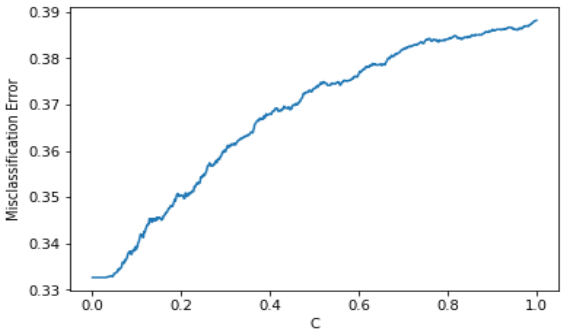

Figure 3: Plot between C and misclassification error clarifying best value of C

So we choose:

```
<Figure size 640x480 with 1 Axes>
Optimal          C:          0.17
0.7398791540785499
```

for this problem, we decided to get `random_state=0, tol=1e-5`

5.1.3 Kernel

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role. For linear kernel the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows: $f(x) = B_{(0)} + sum(a_i * (x, x_i))$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B_0 and a_i (for each input) must be estimated from the training data by the learning algorithm.

5.2 Naive Bayes

Naive Bayes classifier calculates the probabilities for every factor, then it selects the outcome with highest probability.
This classifier assumes the features are independent. Hence the word naive. However it's highly used in the field of text identification.

5.2.1 Bayes Rule

It tells us how often X happens given that Y happens, written P(X|Y), when we know how often Y happens given that X happens, written P(Y|X) , and how likely X and Y are on their own.

$$P(X|Y) = \frac{P(Y|X)P(X))}{P(X)}$$

Naive Bayes is based on the Bayes rule and a set of conditional independence assumptions. Given the goal of learning P(Y|X) where X = (X_1 ...,Xn_i) the Naive Bayes algorithm makes the assumption that each Xi is conditionally independent of each of the other Xks given Y, and also independent of each subset of the other Xk's given Y. The value of this assumption is that it dramatically simplifies the representation of P(X|Y), and the problem of estimating it from the training data.

5.2.2 Multinomial Naive Bayes

The multinomial Naive Bayes classifier is suitable for classification with discrete features. The multinomial distribution works well with TF-IDF as well.

5.3 Decision Tree

A decision tree is a simple representation for classifying examples. Decision tree learning is one of the most successful techniques for supervised classification learning. For this section, assume that all of the features have finite discrete domains, and there is a single target feature called the classification. Each element of the domain of the classification is called a class.
A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.

5.3.1 Gini Index

Gini index (a criterion to minimize the probability of misclassification):

$$Gini = 1 - \sum_j p_j^2$$

The Gini index is maximal if the classes are perfectly mixed.

5.3.2 Entropy

In machine learning sense and especially in this case Entropy is the measure of homogeneity in the data. Its value is ranges from 0 to 1. Its value is close to 0 if all the example belongs to same class and is close to 1 is there is almost equal split of the data into different classes. The formula to calculate entropy is :

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Here p_i represents the proportion of the data with ith classification and c represents the different types of classification.

5.3.3 Information Gain

Now Information Gain measure the reduction by classifying the data on a particular attribute. The formula to calculate Gain:

$$IG(D_p) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

where I could be entropy or Gini index, or classification error, (D_p), (D_{left}), and (D_{right}) are the dataset of the parent, left and right child node.

5.3.4 Parameters

- **Criterion**: The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.
- **max_depth** : The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

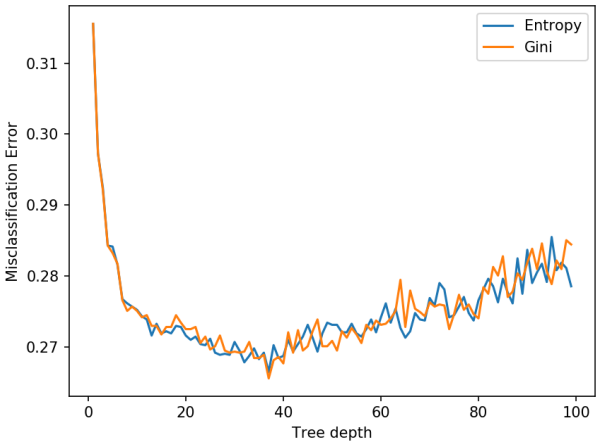

Figure 4: A plot between tree depth and misclassification error showing the best value for tree depth.

5.4 Random Forest Classifier

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object.
Since the prediction of a forest is the average of the predictions of its trees. The average can be written as:

$$F(x) = \frac{1}{J}\sum_{j=1}^{J} f_j(x)$$

Where J is the number of trees in the forest.
The prediction is the average of the bias terms

$$F(x) = \frac{1}{J}\sum_{j=1}^{J} c_{j\,full} + \sum_{k=1}^{K}(\frac{1}{J}\sum_{j=1}^{J} contrib_j(x,k))$$

plus the average contribution of each feature

5.4.1 Random Forest Parameters

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

n_estimators: The number of trees in the forest.

max_depth: The maximum depth of the tree.

min_samples_split:
The minimum number of samples required to split an internal node:

- int: then consider min_samples_split as the minimum number.
- float: then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

min_samples_leaf:
The minimum number of samples required to be at a leaf node. This may have the effect of smoothing the model, especially in regression.

- int: then consider min_samples_leaf as the minimum number.
- float: then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.

max_features:
The number of features to consider when looking for the best split:

- int: consider max_features features at each split.
- float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.
- “sqrt” = “auto”: then max_features = sqrt(n_features).
- “log2”, then max_features = $\log_2$(n_features).
- None, then max_features = n_features.

5.5 Logistic Regression

Logistic Regression (also called Logit Regression) is commonly used to estimate the probability that an instance belongs to a particular class

5.5.1 Estimating Probabilities

Logistic Regression model computes a weighted sum of the input features (plus a bias term), but instead of outputting the result directly like the Linear Regression model does, it outputs the logistic of this result $\hat{p} = h_\theta(\mathbf{x}) = \sigma(\theta^T \cdot \mathbf{x})$

The logistic — also called the logit,is a sigmoid function that outputs a number between 0 and 1

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

Once the Logistic Regression model has estimated the probability that an instance x belongs to the positive class, it can make its prediction ŷ easily

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5, \\ 1 & \text{if } \hat{p} \geq 0.5. \end{cases}$$

5.5.2 Training and Cost Function

The objective of training is to set the parameter vector θ so that the model estimates high probabilities for positive instances (y = 1) and low probabilities for negative instances (y = 0). This idea is captured by the cost function

$$c(\theta) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1, \\ -\log(1-\hat{p}) & \text{if } y = 0. \end{cases}$$

The cost function over the whole training set is simply the average cost over all training instances. It can be written in a single expression (as you can verify easily), called the log loss

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(\hat{p}^{(i)}) + (1-y^{(i)})log(1-\hat{p}^{(i)})]$$

5.5.3 Decision Boundaries

The hyperparameter controlling the regularization strength of a Scikit-Learn LogisticRegression model is not alpha (as in other linear models), but its inverse: C. The higher the value of C, the less the model is regularized.

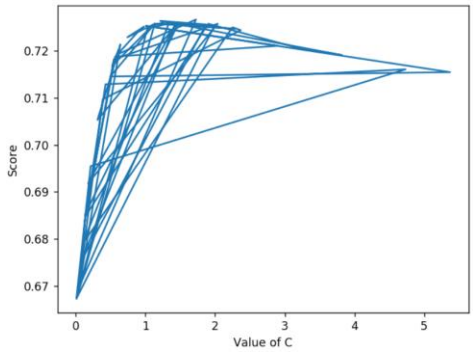

Figure 5: A plot between the value of C and the Score showing the best value of C.

5.6 K nearest Neighbor

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry.

It used Euclidean distance to calculate distance between test instance and other training points.

$d(x,x')$=Sqrt((x1−x′1)2+(x2−x′2)2+…+(xn−x′n)2)

Prediction Equation:

$$y = \frac{1}{K}\sum_{i=1}^{K} y_i$$

5.6.1 Tuning

We tune parameter K to get best accuracy:

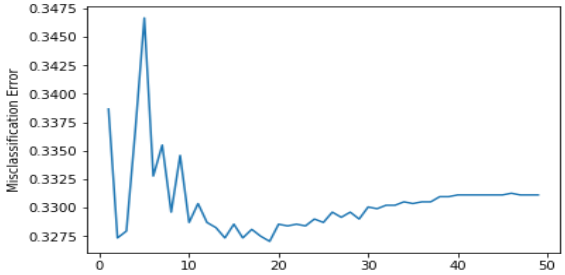

Figure 7: A plot between the value of K and the misclassification error showing the best value for K.

5.7 Voting Classifier

Voting classifier is one of the ensemble methods whose goal is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator. Suppose you have trained a few classifiers and we take the majority of them.

A very simple way to create an even better classifier is to aggregate the predictions of each classifier and predict the class that gets the most votes.

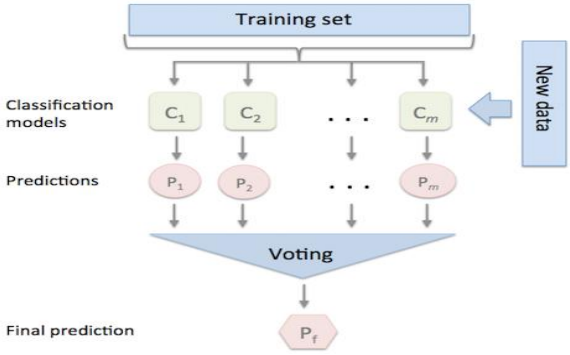

Figure 6: A diagram clarifying how voting classifier works

5.7.1 Soft voting

In soft voting, we predict the class labels based on the predicted probabilities p for classifier where w_j is the weight that can be assigned to the jth classifier.

$$\hat{y} = \arg\max_i \sum_{j=1}^{m} w_j p_{ij}$$

Accordingly, we chose our best performing classifiers to get a more precise result. For the parameter "estimators" we chose Random Forest, Logistic Regression, SVM and Naive Bayes classifiers.

6.Discussion

6.1 K-Fold Cross Validation

In statistics and machine learning we usually split our data into two subsets: training data and testing data (and sometimes to three: train, validate and test), and fit our model on the train data, in order to make predictions on the test data. When we do that, one of two thing might happen: we overfit our model or we underfit our model. We don't want any of these things to happen, because they affect the predictability of our model .

Overfitting means that model we trained has trained too well and is now, well, fit too closely to the training dataset. This usually happens when the model is too complex like
" too many features/variables compared to the number of observations". This model will be very accurate on the training data but will probably be very not accurate on untrained or new data.

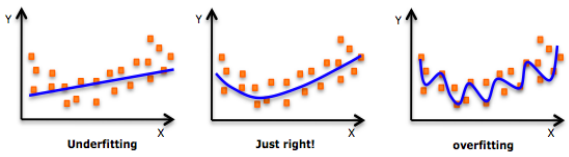

Figure 8: This graph demonstrates underfitting and overfitting

So we use K fold Cross Validation to estimate the skill of a machine learning model on unseen data.

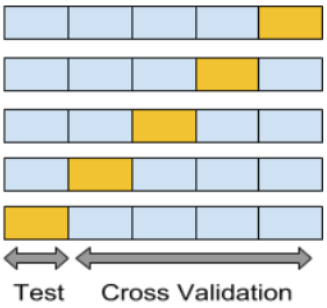

Figure 9: K-fold cross validation example with 5 sections

We tuned our hyperParamters using Kfold cross validation to select the best model avoiding Overfitting problem.

6.2 Classifiers Accuracy

After Tuning we found the best classifiers used for detecting offensive tweets are Random Forest and SVM.
Random Forest is based on Decision Tree concept and one of its best advantages that it work perfect for textual data and if we used it in numerical data (due to TFIDF vectorization) it put ranges & algorithm put cut points are basically value in table and it calculate as positive portion and negative portion after and before cut point so it could calculate entropy.
SVM is based on perceptron concept as it generate several lines and choose the best from them.

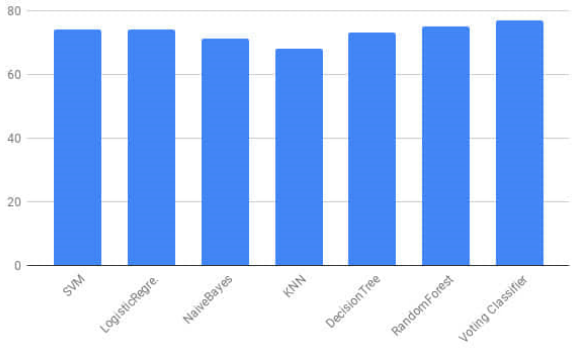

Figure 10: A histogram showing the accuracy of SVM, Logistic Regression, Naive Bayes, KNN, Decision Trees, Random Forests and Voting classifier

6.3 Confusion Matrix

Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.
Thus, here is an example for the accuracy of one of our models:

I'm Assuming I'm Not Going to Get an Answer': Brainless #Liberal Stooge Ocasio-Cortez .@USER on $40T Policy Plan URL #WakeUpAmerica THIS is NOT America's future VOTE Republican or surrender America to these #DEM morons @USER #MAGA TN	@USER @USER Such supposedly smart people didn't think for one moment that in the audience there are people who voted Trump and are scared shitless hearing the top brass speak that way.
@USER And the brainless #DEMS wants us to believe that he doesn't have the majority. Liberals is just like their leaders" the corrupts #Obama and #CrookedHillary both are real bad losers #MAGA #TRUMP"	Nina One of the reasons I didn't read this book for so long was because I was SO WORRIED about her character. I should've trusted everyone. Nina is a goddamn goddess I would die for a thousand times over. She is big and bold and lovely and PERFECT. HER ARC. IS. PERFECT.

Figure 11: Example of the confusion matrix of one of the models

7 Results

Results differ according to the classifier used. The diversity is not a wide one. However, we were able to define the best performing classifiers which will be illustrated using accuracy measure and confusion matrix.

- Voting classifier got the highest score with almost 76%
- Random Forest has the second highest accuracy score reaching almost 74.8%
- Logistic regression comes next with accuracy 74.8%
- Decision Trees and SVM are nearly the same; approaching an accuracy of 73.5%
- Naive Bayes accuracy scored 71%
- K-Nearest Neighbors scored 68.5%

7.1 Analysis

For a better visualization of the goodness of each classifier, listed below are confusion matrices output for the classifiers used.

Predict \ Actual	Positive	Negative
Positive	3804	618
Negative	1200	998

Figure 12: SVM classifier

Predict \ Actual	Positive	Negative
Positive	4270	152
Negative	1578	620

Figure 13: Random Forest classifier

Predict \ Actual	Positive	Negative
Positive	4112	310
Negative	1419	779

Figure 14: Logistic Regression classifier

Predict \ Actual	Positive	Negative
Positive	4363	59
Negative	1875	323

Figure 15: Naive Bayes classifier

Predict \ Actual	Positive	Negative
Positive	4414	8
Negative	2146	52

Figure 16: KNN classifier

Predict \ Actual	Positive	Negative
Positive	4172	250
Negative	1556	642

Figure 17: Decision Tree classifier

Predict \ Actual	Positive	Negative
Positive	858	26
Negative	294	146

Figure 18: Voting classifier

8 References

[1] Hands on Machine Learning,Aurélien Géron

[2] Random Forests, Leo Breiman ,Statistics Department University of California Berkeley, CA 94720

[3] Introduction to Logistic Regression Analysis and Reporting, CHAO-YING JOANNE PENG KUK LIDA LEE GARY M. INGERSOLL, Indiana University- Bloomington

[4] Introduction to Decision Trees, J.R. QUINLAN ,New South Wales Institute of Technology

[5] Using TF-IDF to Determine Word Relevance in Document Queries, Juan Ramos, Rutgers University

[6] Language Normalization Of Noisy Text Data, AKASH MOHANTY,SANTOSH KUMAR MAJHI,SAMPA CHAUPATTNAIK, Utkal University