**Ads classification**

Aditya Kanungo  
2nd June 2018

**Contents**

**1 Introduction**  
 1.1 Problem Statement   
 1.2 Data

**2 Methodology**

2.1 Pre-Processing

2.1.1 Missing value-analysis  
 2.1.2 General text comments study  
 2.1.3 Text Data Cleaning  
 2.1.4 Word Cloud to category wise visualize frequent comments  
  
 2.2 Vectorization of data  
 2.2.1 Result of vectorizing train dataset  
 2.2.2 Result of vectorizing test dataset  
  
 2.3 Modeling  
 2.3.1 Model Selection  
 2.3.2 Binary classification

**3 Conclusion**

3.1 Model Evaluation

**Appendix – Python Code  
References**

**Chapter 1**

**Introduction**

* 1. **Problem Statement**

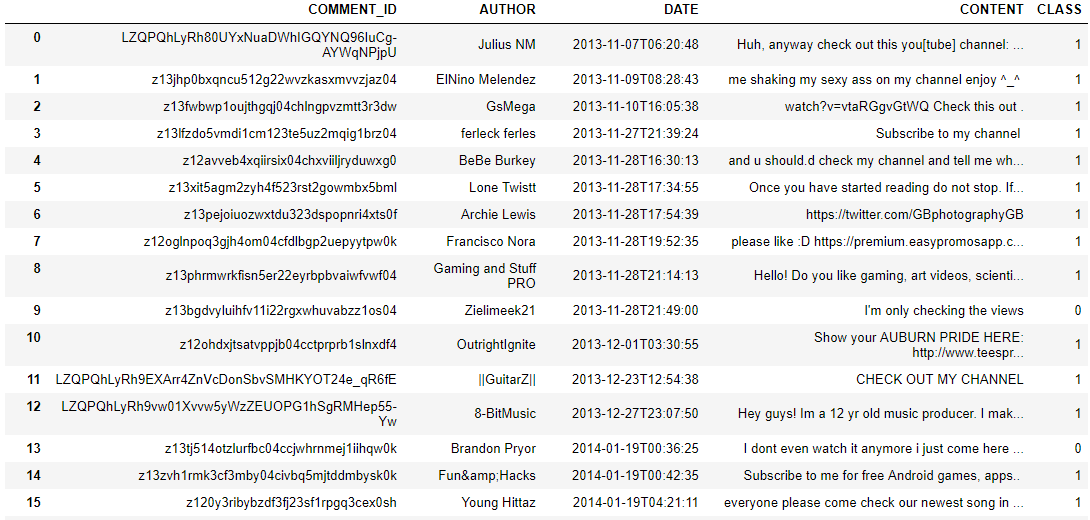
You know much we all love internet. We connect, learn and have fun through this window to the world. It's nearly free and is become more and more a community thing. Platforms struggle to effectively facilitate conversations, leading many communities to limit user experience. However, just like we take care of our communities, we need to keep internet clean and fresh. One of the things you all might have come across is random comments and remarks on articles, blogs, videos. While everyone is free to do what they want, random and unwanted ads take the experience away.

In this dataset we have 5 topics on which people commented. A lot of them were advertisements. When a article becomes popular, and draws people it also draws these unwanted adverts in comments. Our task is to predict whether a comment is an advertisement or not. We’ll be using dataset 5 topics on which people commented. A lot of them were advertisements. When an article becomes popular and draws people it also draws these unwanted adverts in comments.

* 1. **Data**

Our task is to build a model which will predict the probability of a comment to be under the different types of toxicity like threats, obscenity, insults, and identity-based hate.

* + 1. **Train Dataset:** (AUTHOR, DATE, CONTENT, CLASS)  
       (1157 rows × 4 columns)



* + 1. **Test Dataset:** (ID, AUTHOR, DATE, CONTENT)

(799 rows × 4 columns)



* + 1. **Sample submission:** (COMMENT\_ID, CLASS)



**Chapter 2**

**Methodology**

**2.1 Pre-Processing**

Before implementing any Machine Learning model on our training data, we are required to look at the data before we start modeling. However, in data science terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. In our case we have text data of 5 topics on which people commented. A lot of them were advertisements.

**2.1.1 Missing value-analysis**

We will first check for any missing values in the comments to check if any comment is null or empty. During which we found that there are 138 missing values under date feature of train data-set and 107 missing values under date feature of test data-set.

Train data-set:

COMMENT\_ID 0

AUTHOR 0

DATE 138

CONTENT 0

CLASS 0

dtype: int64

Test data-set:

ID 0

COMMENT\_ID 0

AUTHOR 0

DATE 107

CONTENT 0

dtype: int64

**2.1.2 General text comments study**

We will do a general text comments study to understand the data text data better, below are the results of that study.

2.1.2.1 All comments

Total no. of comments/CONTENT in train data: 1157

Total no. of ads as comments : 586

Percentage of ads as comments : 50.648228176318064

Percentage of real comments : 49.351771823681936

Total number of NAN/Null comments : 0

2.1.2.2 Category wise comments/CONTENT

Total no. of ads as comments in train data : 586

Total no. of real comments in train data : 571

2.1.2.3 Datatype of train data

COMMENT\_ID object

AUTHOR object

DATE object

CONTENT object

CLASS int64

dtype: object

2.1.2.4 Average, min and max length of comments/CONTENT

Mean length of comment : 94.25324114088158

Min length of comment : 2

Max length of comment : 1200

**2.1.3 Text Data Cleaning**

One of the first steps in working with text data is to clean the data. It is an essential step before the data is ready for analysis. Majority of available text data is highly unstructured and noisy in nature – to achieve better insights or to build better algorithms, it is necessary to play with clean data. For example, social media data is highly unstructured – it is an informal communication – presence of unwanted content like Numbers, Stopwords, punctuations, whitespaces, uppercase etc. are the usual suspects.

We took the following steps to clean our text data:

General cleaning:

* Converting uppercase to lowercase.
* Replacing ‘s with blank.
* Replacing few common abbreviations with proper words.
* Removing non-alphanumeric characters.
* Removing white spaces
* Removing \_ (underscore)
* Removing English stop words
* Removing numeric values

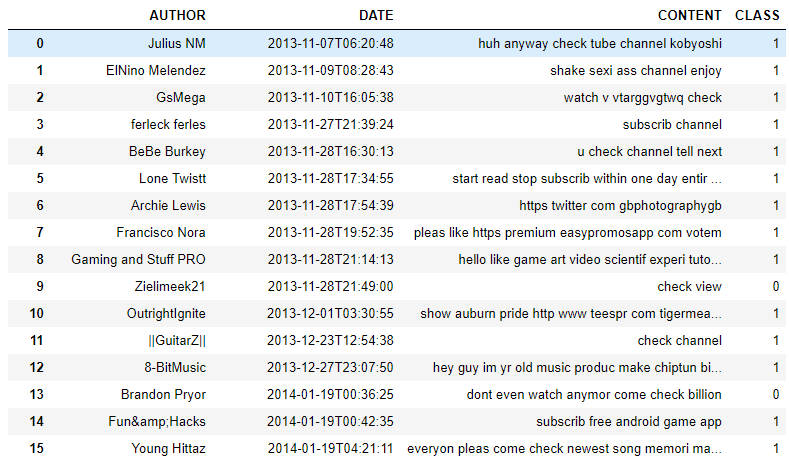
Lemmatization:

* Lemmatization brings a word down to its actual base form which, in the case of irregular verbs, might look nothing like the input word
* E.g. - geese to goose and meanness and meaning

Stemming:

- Stemming is use to get all the different forms of the same word down to a base from which   
 need not be a legit word on its own  
- E.g. - apple and apples down to appl, and it stems berry and berries to berri.

2.1.4.1 Train dataset after data cleaning

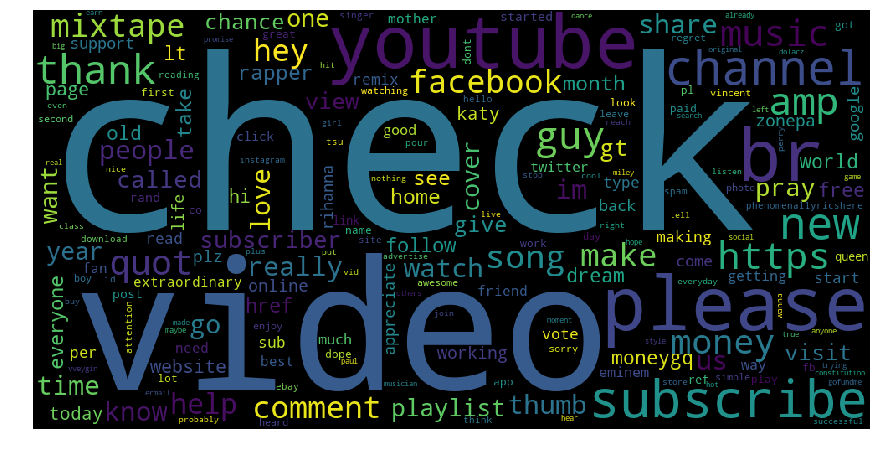


2.1.4.2 Test dataset after data cleaning

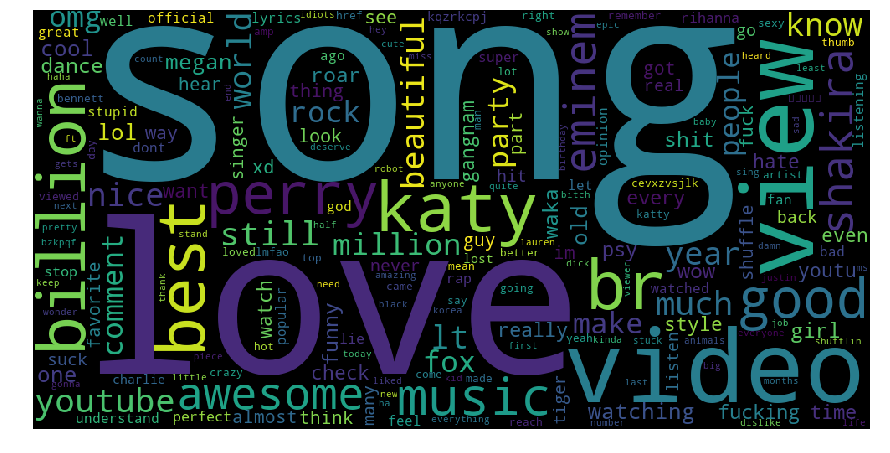


**2.1.4 Word Cloud to category wise visualize frequent comments**

**2.1.4.1 Word cloud for Ads comments**

****

**2.1.4.2 Word cloud for real/actual comments**

****

**2.2 Vectorization of data**

After cleaning the text data, it is needed to transform text into something a machine can understand before feeding it to a machine learning model. That is, transforming text into a meaningful vector (or array) of numbers.

In our model we will be using TF-IDF (term frequency–inverse document frequency) method for vectorization. It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. tf-idf is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf-idf.

2.2.1 Result of vectorizing train dataset:

<1157x2374 sparse matrix of type '<class 'numpy.float64'>'

with 8687 stored elements in Compressed Sparse Row format>

2.2.2 Result of vectorizing test dataset:

<799x2374 sparse matrix of type '<class 'numpy.float64'>'

with 4963 stored elements in Compressed Sparse Row format>

**2.3 Modeling**

**2.3.1 Model Selection**

In early stage of our analysis process we have come to understand that our most important feature is the text CONTENT will have all the comments.

Therefore, before moving for further analysis we will make sure our comments/CONTENT is clean and thus we will proceed with text cleaning operations.

After that we will vectorize the data into document-term-matrix and learn the vocabulary of our train data to make a model on top of it and compare the output with the actual values of train data to compute the training accuracy of our model.

After that we will implement the trained model on test data and store the predicted value in final\_submission file as per the sample submission file.

**2.3.2 Binary Classification**

Binary or binomial classification is the task of classifying the elements of a given set into two groups (predicting which group each one belongs to) on the basis of a classification rule.

The actual output of many binary classification algorithms is a prediction score. The score indicates the system’s certainty that the given observation belongs to the positive class. To make the decision about whether the observation should be classified as positive or negative, as a consumer of this score, you will interpret the score by picking a classification threshold (cut-off) and compare the score against it. Any observations with scores higher than the threshold are then predicted as the positive class and scores lower than the threshold are predicted as the negative class.

The predictions now fall into four groups based on the actual known answer and the predicted answer: correct positive predictions (true positives), correct negative predictions (true negatives), incorrect positive predictions (false positives) and incorrect negative predictions (false negatives). Which we can see in our Confusion Matrix.

**2.3.3 Logistic regression**

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). Rather than choosing parameters that minimize the sum of squared errors (like in ordinary regression), estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values.

**2.3.4 Gaussian Naive Bayes Classifier**

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. Naïve Bayes classifier is one of the most effective machine learning algorithms implemented in machine learning projects and distributed MapReduce implementations leveraging Apache Spark. Primarily Naïve Bayes is a linear classifier, which is a supervised machine learning method and works as a probabilistic classifier as well.

**2.3.5 k-Nearest Neighbors**

the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

**Chapter 3**

**Conclusion**

**3.1 Model Evaluation**

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Ads classification, the latter two, Interpretability and Computation Efficiency, do not hold much significance. Therefore, we will use Predictive performance as the criteria to compare and evaluate models. Predictive performance can be measured by comparing Predictions of the models with real values of the target variables and calculating accuracy or some average error measure.

In our case of Ads classification, we have applied four models namely logistic regression, naïve Bayes, KNN and random forest.

The training accuracy of each is as follows:

Logistic regression : 0.9619706136560069

KNN : 0.7139152981849611

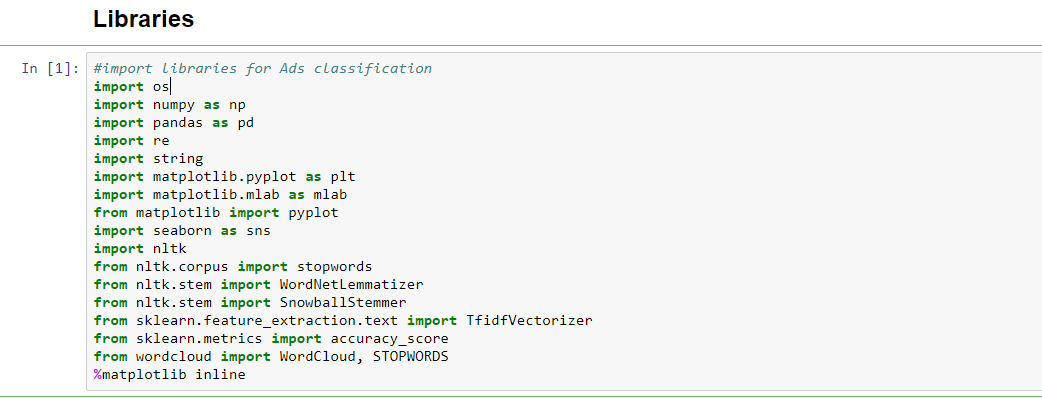
Gaussian Naïve Bayes : 0.9196197061365601

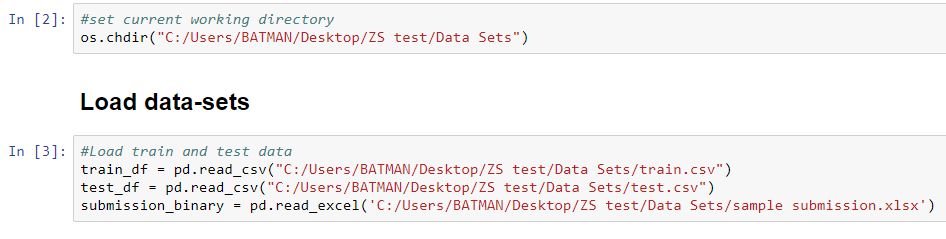
Random Forest : 0.9982713915298185

As per the training accuracy we will select Random forest.

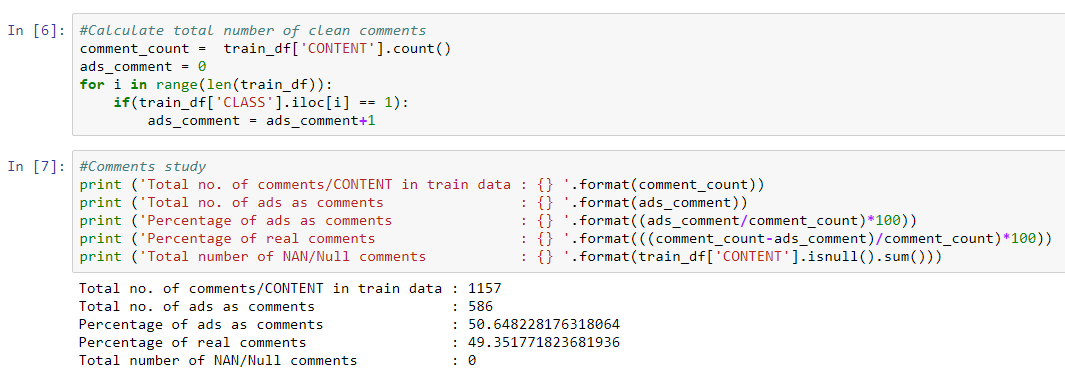
**Appendix - Python Code**

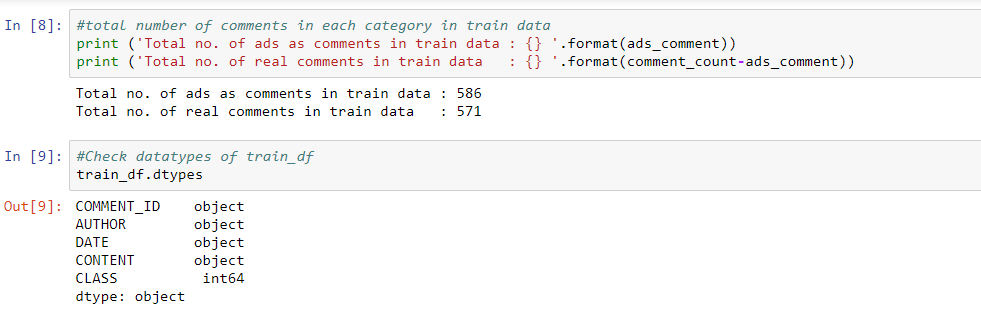
**(Along with Output)**



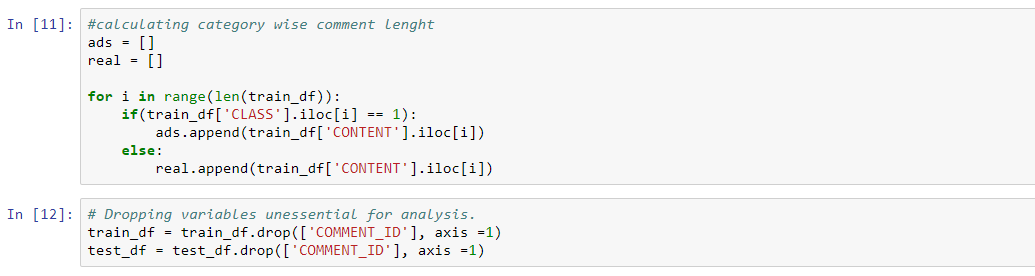


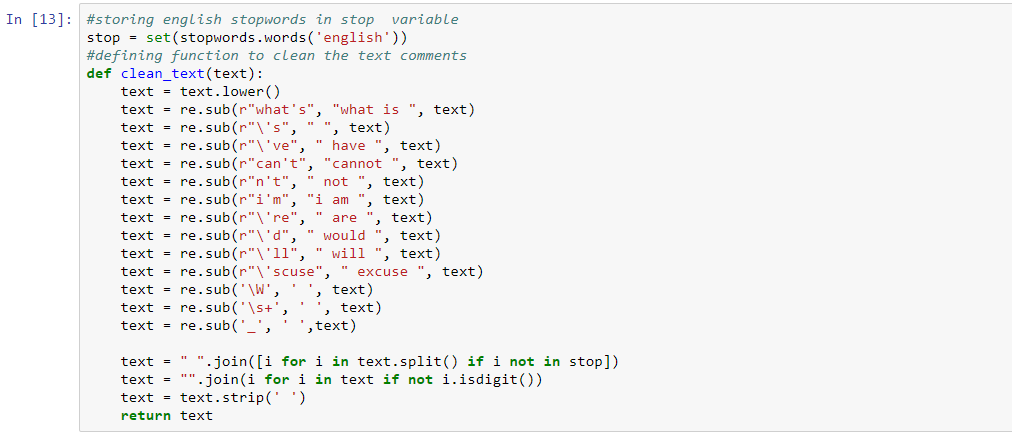


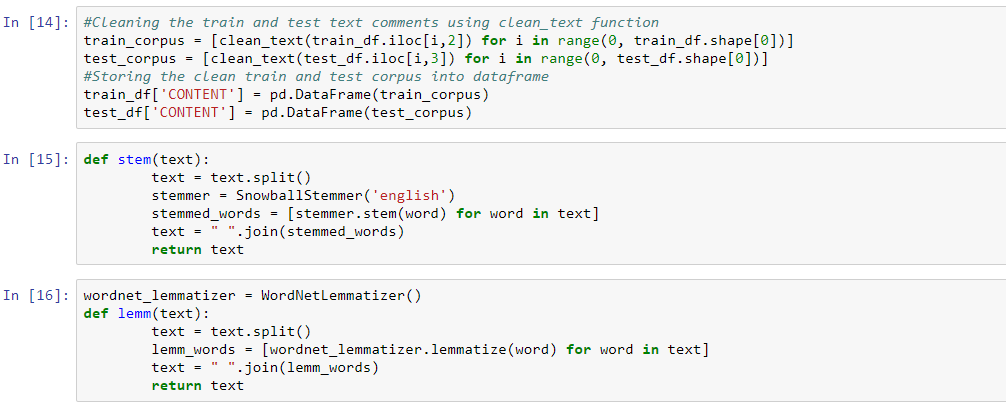


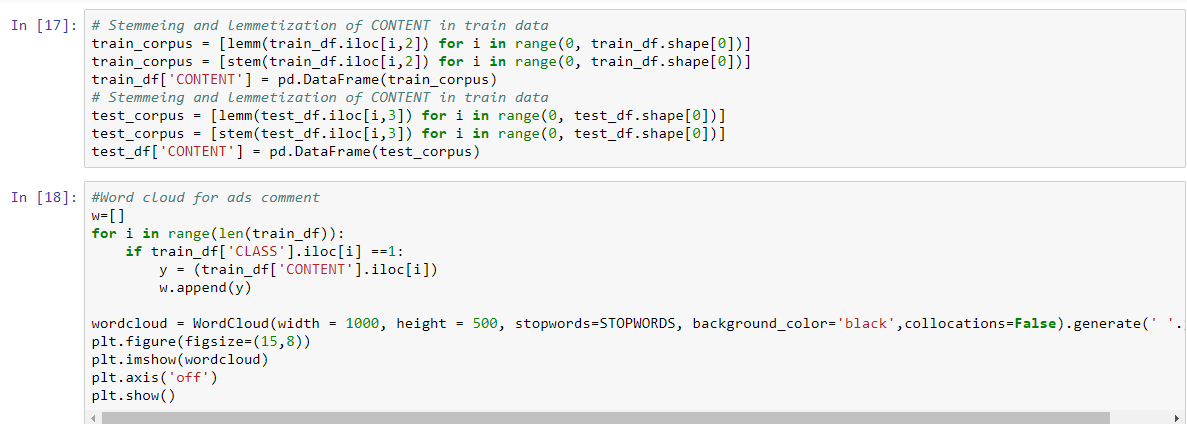


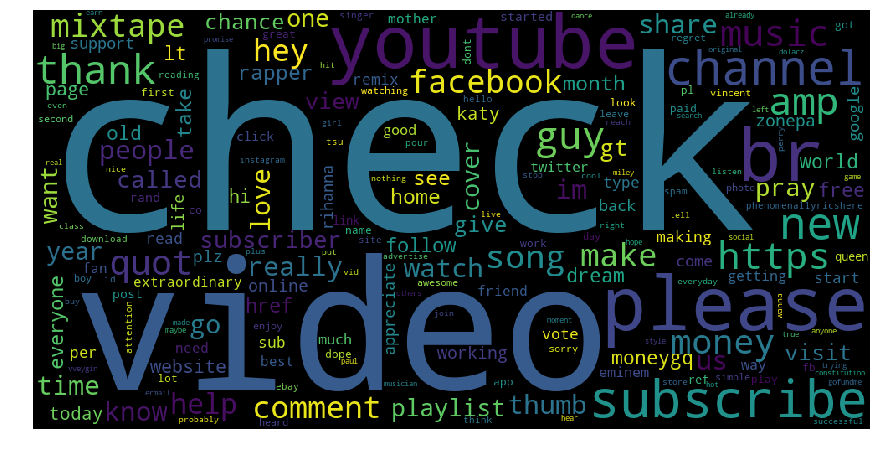


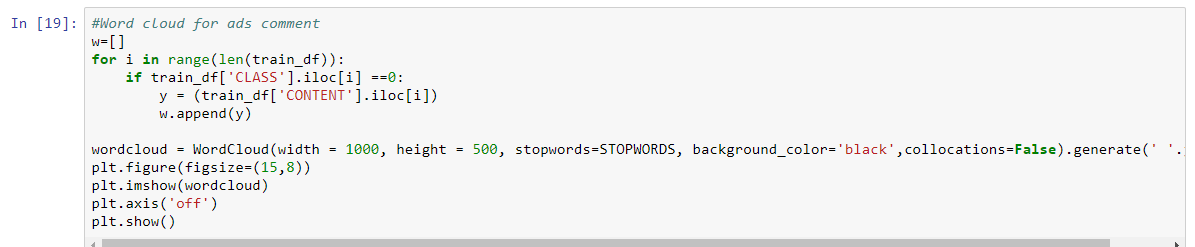


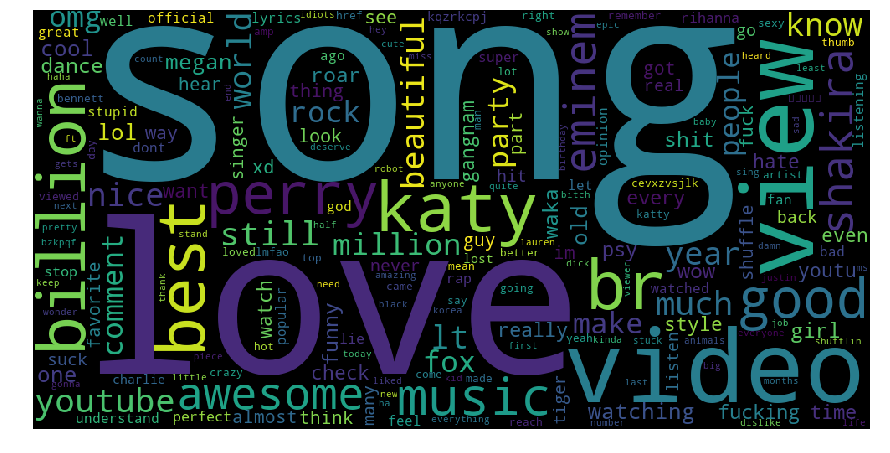






****

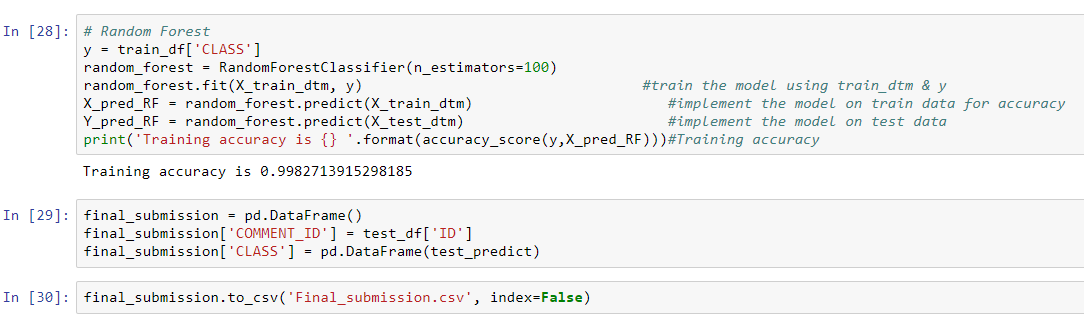


****









**References**

[**https://www.analyticsvidhya.com/**](https://www.analyticsvidhya.com/)

[**https://machinelearningmastery.com**](https://machinelearningmastery.com)

[**https://docs.python.org/3/**](https://docs.python.org/3/)