

**SMS SPAM Detection Classification Project**

Submitted by:

MAHESH AGASHIMANI

**ACKNOWLEDGMENT**

I take great pleasure to thank and acknowledgement the allowance by Data Trained Education and permission by Flip Robo. I extend whole hearted thanks to them I worked and learned a lot and sharing me the knowledge and experience.

Data Trained Education and Flip Robo provided training is the very important to completion of project.

**INTRODUCTION**

* Business Problem Framing

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam.

* Conceptual Background of the Domain Problem

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the nonspam texts. It uses a binary type of classification containing the labels such as ‘ham’ (nonspam) and spam. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails in order to prevent them from getting into the user’s inbox.

* Review of Literature

The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

* Motivation for the Problem Undertaken

A collection of 5573 rows SMS spam messages was manually extracted from the Grumbletext Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages.

A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available.

**Analytical Problem Framing**

* Mathematical/ Analytical Modeling of the Problem

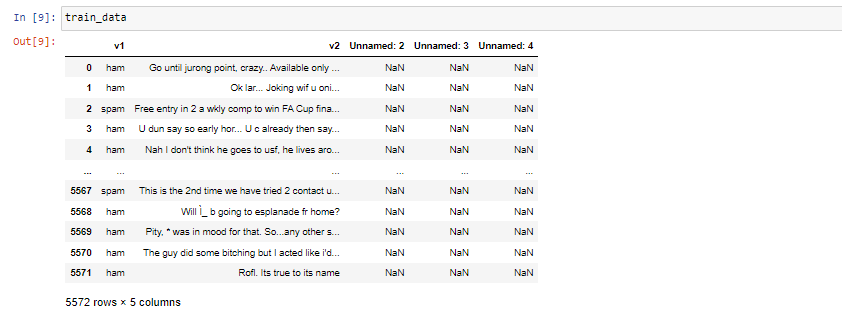
The main goal of these two parts of article is to show how you could design a spam filtering system from scratch.

* Data Sources and their formats

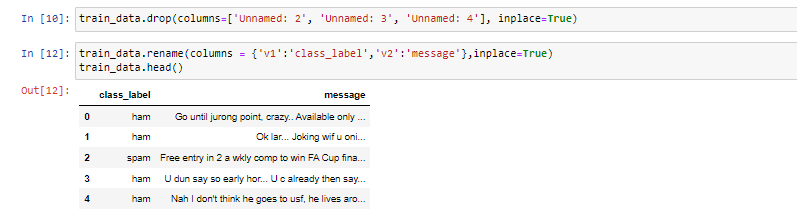
The datasets contain 5574 messages with respective labels of spam and ham (legitimate). With this data, we will train a machine learning model that can correctly classify SMS as ham or spam.

#### Import Libraries and Data:

The spam dataset located in the dataset directory named spam.csv can be imported as follows-

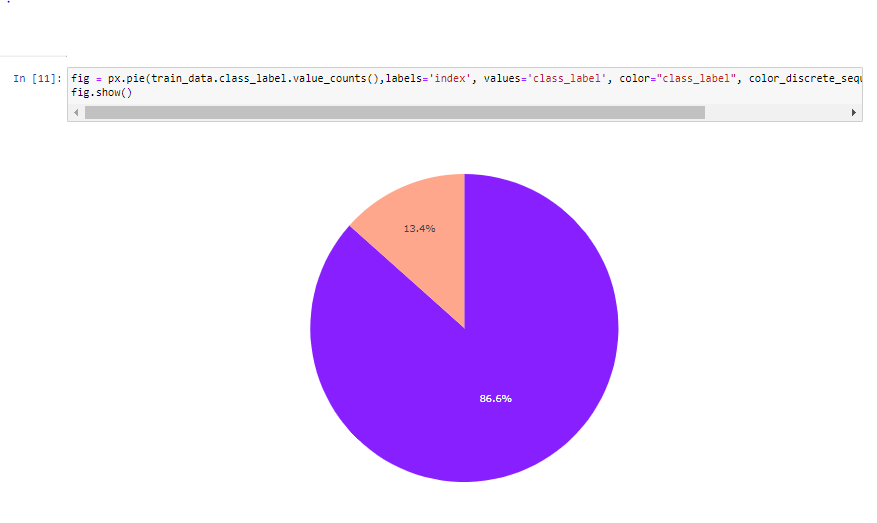


The dataset contains 5 columns. Column v1 is the dataset label (“ham” or “spam”) and column v2 contains the text of the SMS message. Columns “Unnamed: 2”, “Unnamed: 3”, and “Unnamed: 4” contain "NaN" (not a number) signifying missing values. They are not needed, so they can be dropped as they are not going to be useful in building the model.

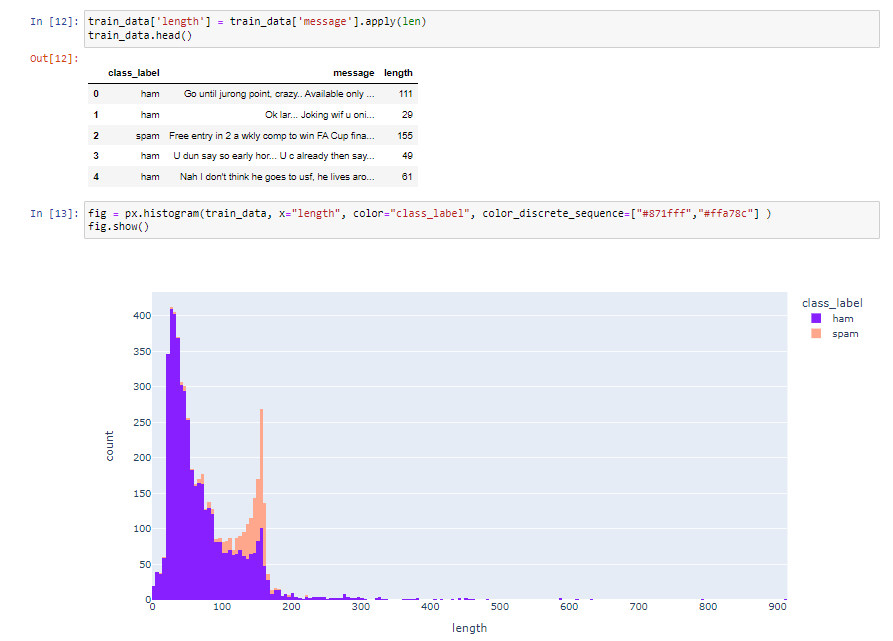




We have an imbalanced dataset, with 747 messages being spam messages and 4825 messages being ham.



The spam makes up 13.4% of the dataset while ham composes 86.6% of the dataset.

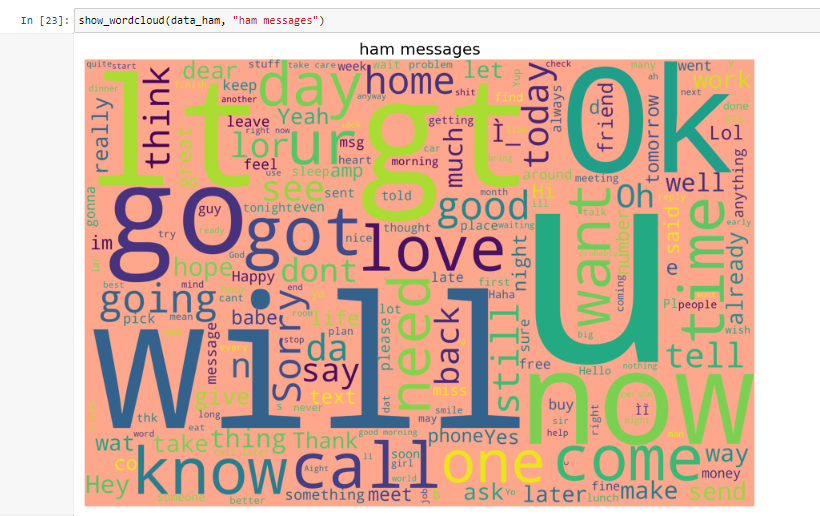


It can be seen that ham messages are shorter than spam messages as the distribution of ham and spam message lengths are centered around 30-40 and 155-160 characters, respectively.

Having a view of the most common words used in spams and hams will help us understand the dataset better. A word cloud can give you an idea of what kind of words are dominant in each class.

To make a word cloud, first separate the classes into two pandas data frames and add a simple word cloud function, as shown below:





* State the set of assumptions (if any) related to the problem under consideration

As given in datasets my assumption is predicting spam or ham messages.

* Hardware and Software Requirements and Tools Used

The needed time to train the model depends on the capability of the used system during the experiment. Some libraries use GPU resources over the CPU to take a shorter time to train a model.

|  |  |
| --- | --- |
| Operating System | Windows 10 |
| Processor | Core i7 |
| RAM | 16GB |
| Graphics card | 1080 TI OC |

Also we are using Jupiter notebook for running the code.

**Model/s Development and Evaluation**

**Preprocess the Data–**

The process of converting data to something a computer can understand is referred to as pre-processing. In the context of this article, this involves processes and techniques to prepare our text data for our machine learning algorithm.

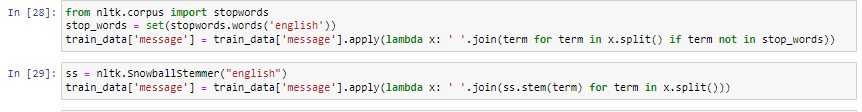
we’ll convert the label to numeric form-



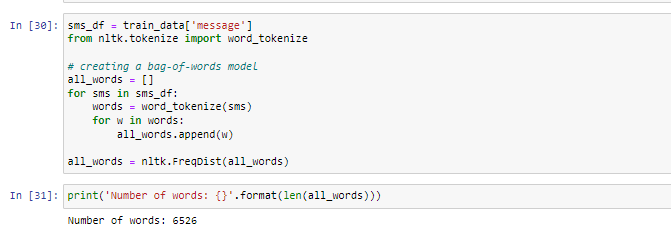
we will process the message content with Regular Expressions (Regex) to keep email and web addresses, phone numbers, and numbers uniform, encode symbols, remove punctuation and white spaces, and finally convert all text to lowercase:



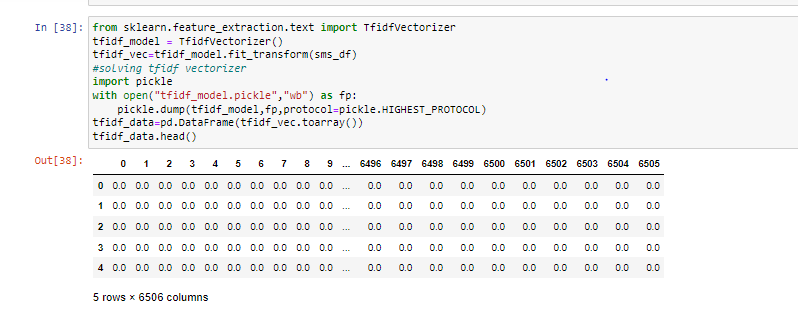
we'll remove stopwords from the message content. Stop words are words that search engines have been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query such as “the”, “a”, “an”, “in”, "but", "because" etc



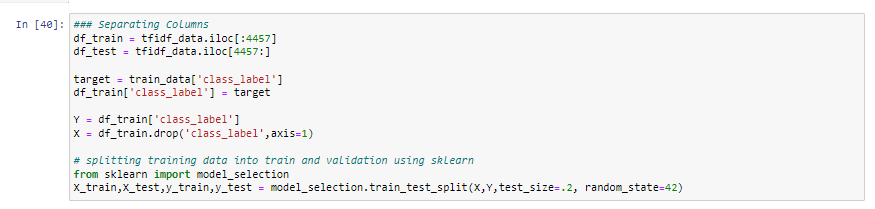
Machine learning algorithms cannot work with raw text directly. The text must be converted into numbers—more specifically, vectors of numbers. Let's split the messages (text data in sentences) into words. This is a requirement in natural language processing tasks where each word needs to be captured and subjected to further analysis.



 The tfidf\_model created from this NLP technique-

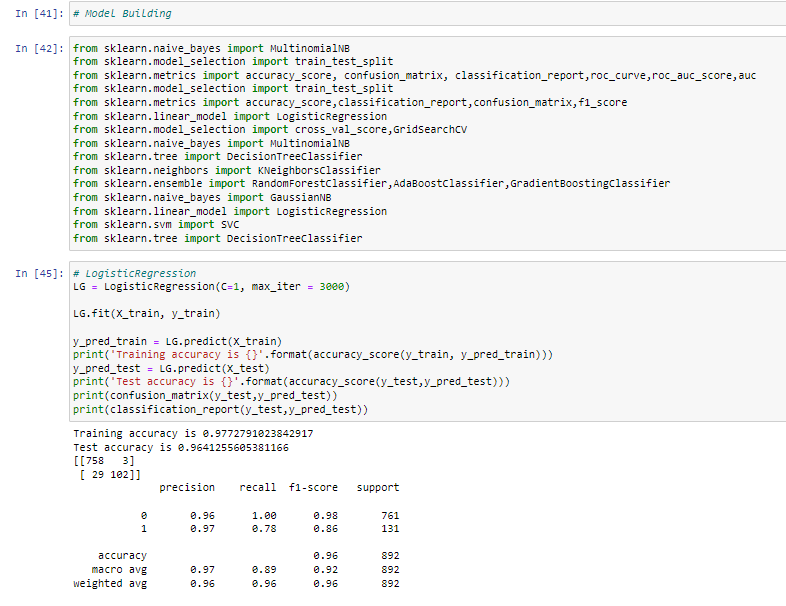


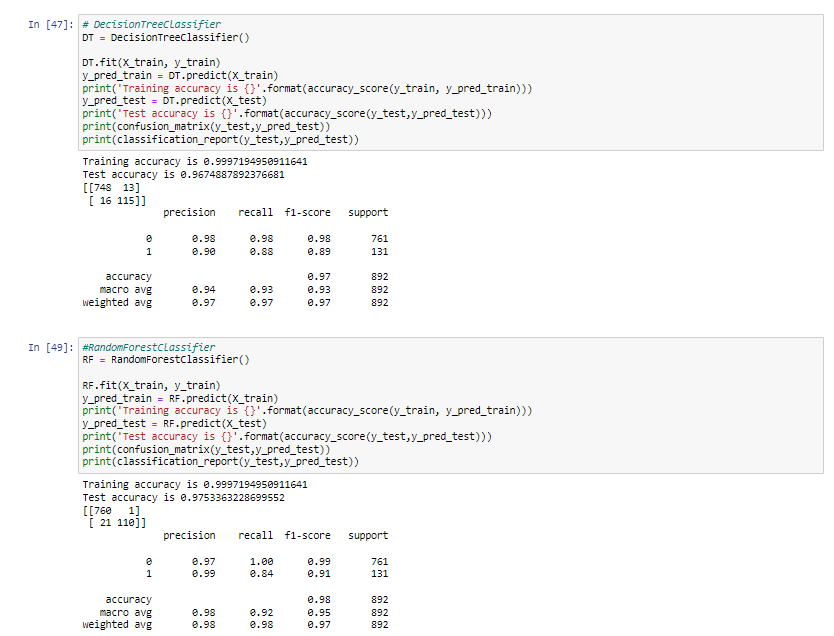
The shape of the resulting dataframe is 5572 by 6506. In order to train and validate the performance of our machine learning model, we need to split the data into training and test dataset respectively. The training set should be later split into a train and validation set.

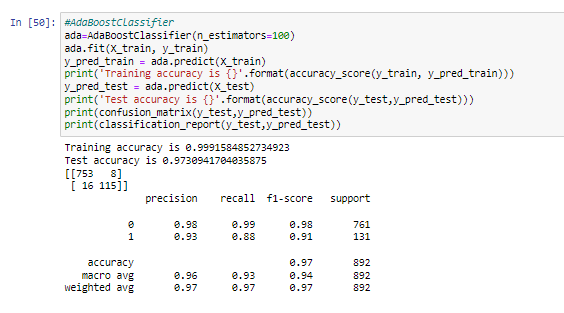


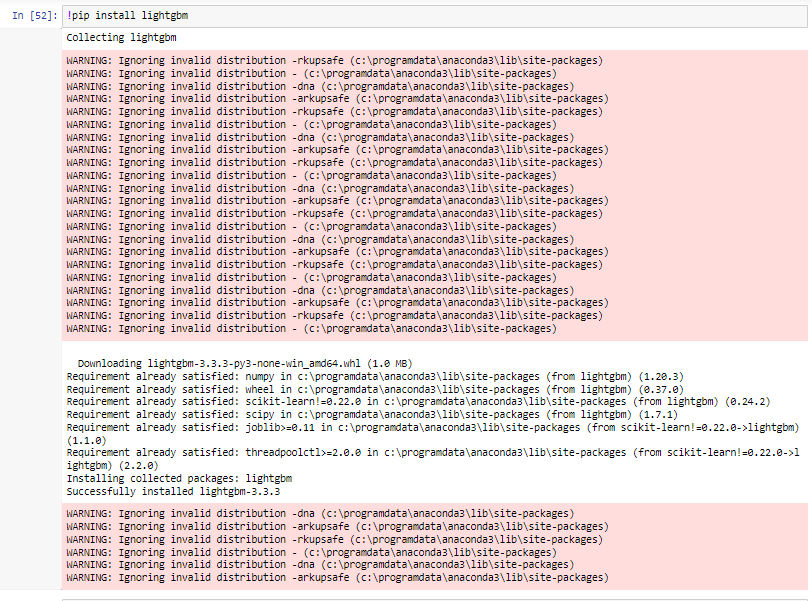
**Model Building–**

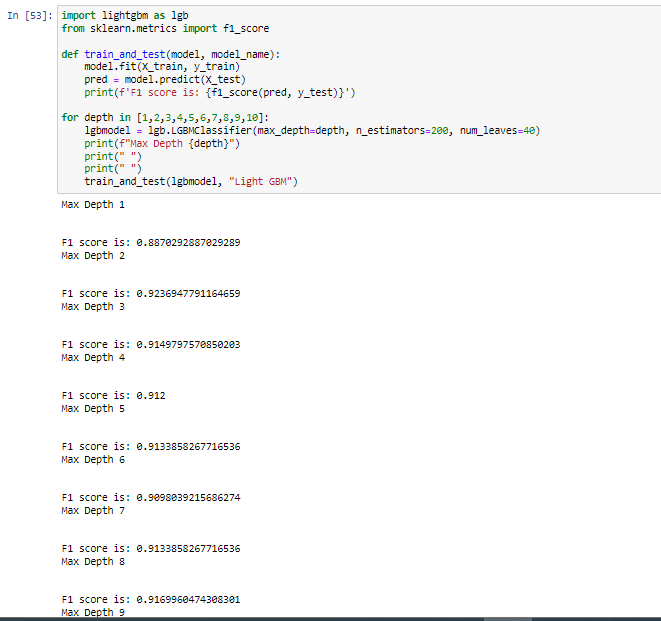
The performance metric for this project is the F1 score. This metric considers both precision and recall to compute the score.















F1 Score is :- 0.891566

**CONCLUSION**

