**SMS Spam Filtering System**

Report

Yixuan Zhu

Mercy College

Term team Project IASP 500

Dr. Zhixiong Chen

December 15, 2016

**Table of Contents**

[Abstract 3](#_Toc469738632)

[Introduction 4](#_Toc469738633)

[Transform Data 4](#_Toc469738634)

[Vocabulary list 5](#_Toc469738635)

[Naive Bayes 5](#_Toc469738636)

[Main Idea 5](#_Toc469738637)

[Result: 6](#_Toc469738638)

[SVM(Support Vector Machine) 7](#_Toc469738639)

[Main Idea 7](#_Toc469738640)

[Result 8](#_Toc469738641)

[KNN (k-Nearest Neighbor) 11](#_Toc469738642)

[Main Idea 11](#_Toc469738643)

[Result: 12](#_Toc469738644)

[Conclusion 14](#_Toc469738645)

[Run the Code 16](#_Toc469738646)

# Abstract

This report will introduces several algorithms and their main idea, how these algorithms use in SMS spam filtering system. All the algorithm in this report use Python to create and base on these package: numpy, sklearn. At last, there will have a conclusion about how these algorithms works in SMS spam filtering system.

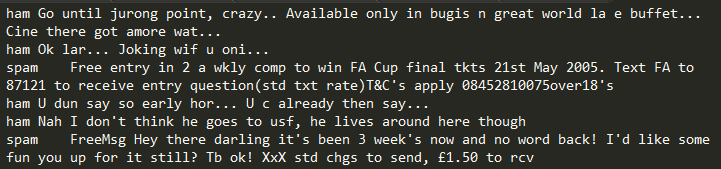
# Introduction

Nowadays, smartphone become an important part in our life. However, system on smartphone is not security enough. In some way, attackers can get your security information, even your bank account through the smartphone. And SMS is one of the most popular way. Some spam SMS with tempting information and an URL, if you click it fall into the trap. So I want find some way to classify the Spam SMS and prevent ourselves from this kind of attack.

# Transform Data

All the data I use is come from UCI Machine Learning Repository. Here is the link: <https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>

The data I have is like this:



First step is separate it, get the SMS label (ham or spam) and all the vocabulary in the sentence. My main idea is, if this SMS is a spam, there must some special words in the sentence to make it more like a spam. Thus, I don’t need the relation between each word, and I separate the sentence as a list of vocabulary.

## Vocabulary list

Computer still cannot calculate on the list of vocabulary. So I put these training data lists together to create a vocabulary list. Every vocabulary will recorded on vocabulary list one time and with the frequency about the vocabulary in the sentence, compare the vocabulary list I can get a vector. Thus, every sentence can transform into vector and it is easy for computer to calculate.

Based on 60 training data, vocabulary has more than 600 lengths and every sentence will transform to 600 lengths vector. If I increase the training data to 120, vocabulary will increase to more than 1000 lengths.

# Naive Bayes

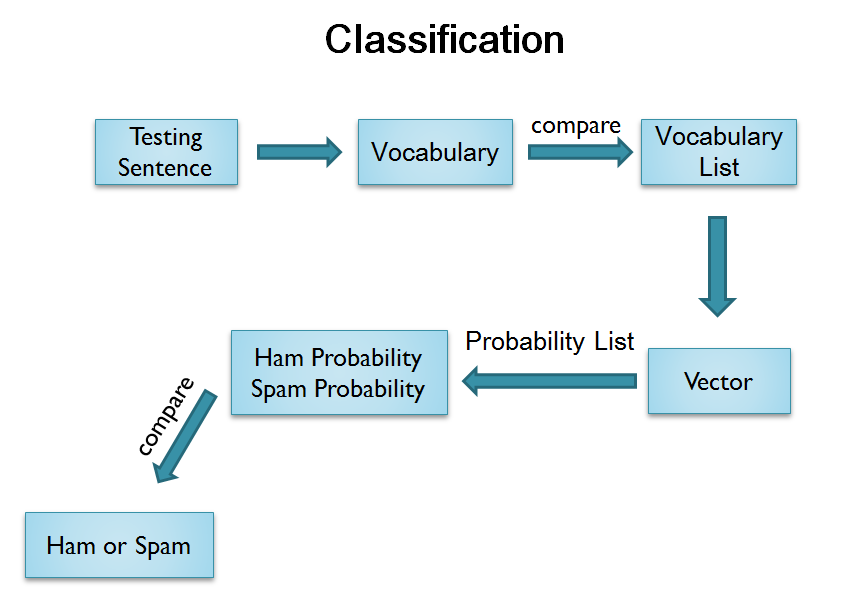
## Main Idea

Naïve Bayes is one of the most popular classification algorithms for spam filtering. It’s base on Bayes' theorem:



P (A) and P (B) are the probabilities of observing A and B without regard to each other. P (A | B), a conditional probability, is the probability of observing event A given that B is true. P (B | A) is the probability of observing event B given that A is true.

According to this formula, I can use train data to create a probability list base on vocabulary list. It means every vocabulary has two probabilities, spam probability and ham probability. When to classify a SMS is spam or not, transform sentence to vector by vocabulary list, then calculate with spam probability and ham probability, get the higher probability. That’s how Naïve Bayes classify SMS data, and here is a flow chart:



## Result:

Based on 60 training data and 20 testing data:

MaxRecall: 0.8

MaxPrecision: 1.0

MinError: 0.2

AveRecall: 0.7068421052631579

AvePrecision: 0.9854166666666666

AveError: 0.3

AveTime: 0.028286582971390355

Based on 120 training data and 40 testing data:

MaxRecall: 0.8974358974358975

MaxPrecision: 1.0

MinError: 0.125

AveRecall: 0.7921558704453442

AvePrecision: 0.9815240111195994

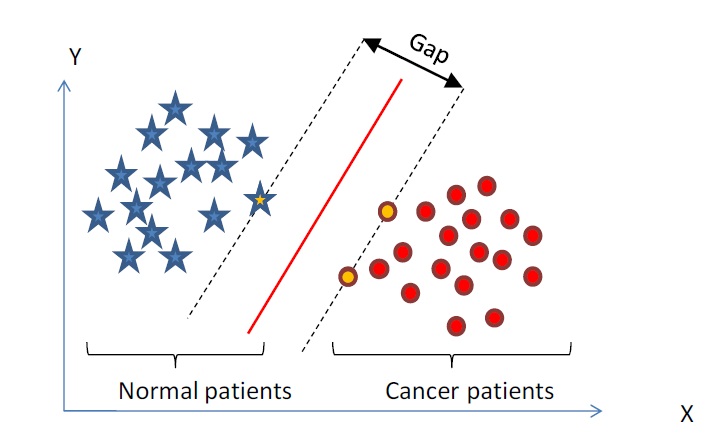
AveError: 0.22000000000000003

AveTime: 0.08877316643805747

# SVM(Support Vector Machine)

## Main Idea

SVM is one of the most popular algorithms in machine learning, it’s powerful and has high accuracy. SVM’s main idea is draw a line to classify data.



If there have two kinds of data like this picture, we can draw unlimited numbers of line to separate it. But there must have a line that can make the biggest gap, this line called support vector. If data cannot be classified by a line, SVM can use kernel to create dimensions and draw the line on the high dimension.Fa

## Result

SVM algorithm in my project is based on sklearn package.

Based on 60 training data and 20 testing data:

Default(C=1 Kernel=‘rbf’):

MaxRecall: 1.0

MaxPrecision: 1.0

MinError: 0.05

AveRecall: 0.6505847953216375

AvePrecision: 0.9140598290598291

AveError: 0.425

AveTime: 0.06084513318737096

Increase the penalty parameter C(C=10.0 Kernel=‘rbf’):

MaxRecall: 0.9444444444444444

MaxPrecision: 1.0

MinError: 0.1

AveRecall: 0.8248402869919899

AvePrecision: 0.9303405572755418

AveError: 0.23500000000000004

AveTime: 0.04786081637264809

Choose a better kernel(C=10.0 Kernel=‘linear’):

MaxRecall: 0.9

MaxPrecision: 1.0

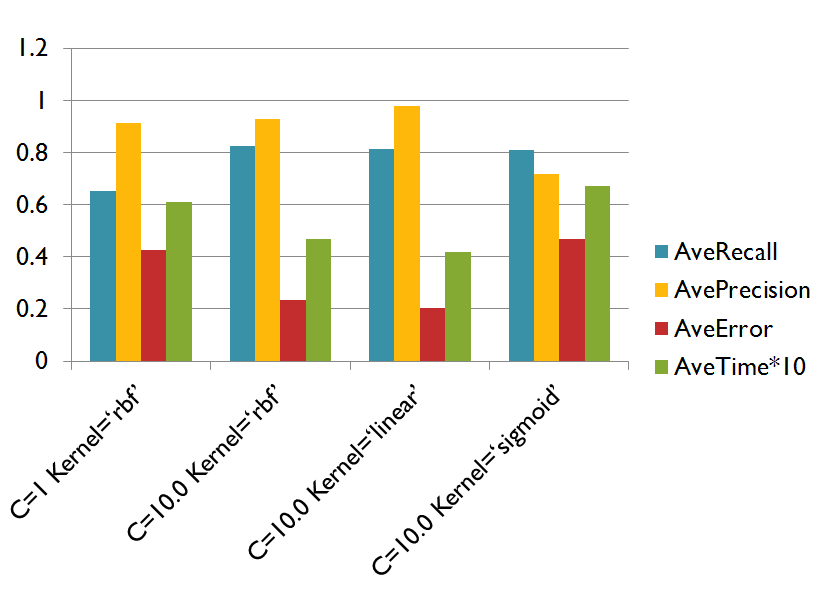
MinError: 0.1

AveRecall: 0.8122222222222222

AvePrecision: 0.977124183006536

AveError: 0.205

AveTime: 0.04208579490100922



Based on 120 training data and 40 testing data:

Default(C=1 Kernel=‘rbf’):

MaxRecall: 1.0

MaxPrecision: 1.0

MinError: 0.275

AveRecall: 0.7568589743589743

AvePrecision: 0.7291666666666667

AveError: 0.5125000000000001

AveTime: 0.19078213592662593

After improve(C=10.0 Kernel=‘linear’):

MaxRecall: 1.0

MaxPrecision: 1.0

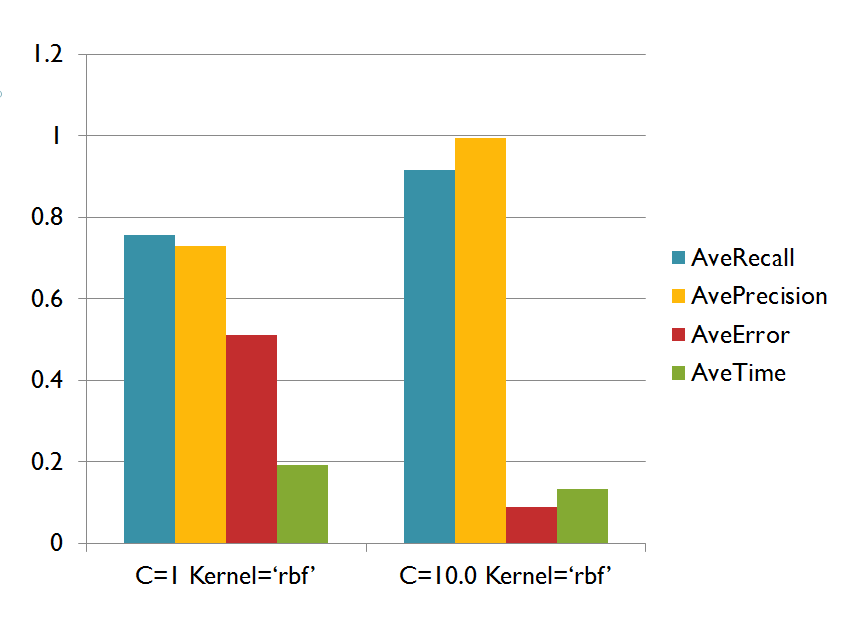
MinError: 0.0

AveRecall: 0.9170512820512821

AvePrecision: 0.9945112781954887

AveError: 0.08750000000000001

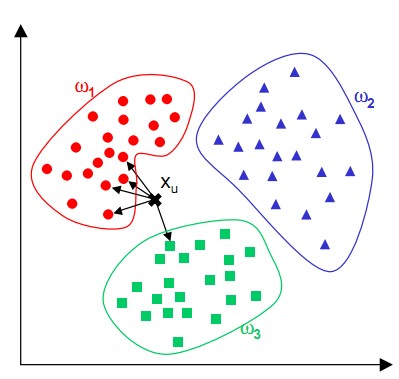
AveTime: 0.13327726086104563



# KNN (k-Nearest Neighbor)

## Main Idea

According to attributes of the data, each data can be located on an axis. To classify a testing data by KNN, just need to locate all the training data on the axis, and find several nearest data around testing data. Thus, maximum number of class and it’s the class of testing data.



## Result:

KNN algorithm in my project is based on sklearn package.

Based on 60 training data and 20 testing data:

Default (n\_neighbors=5):

MaxRecall: 0.6

MaxPrecision: 1.0

MinError: 0.4

AveRecall: 0.495

AvePrecision: 1.0

AveError: 0.505

AveTime: 0.060209280330714585

After improve (n\_neighbors=3,algorithm='ball\_tree'):

MaxRecall: 0.65

MaxPrecision: 1.0

MinError: 0.35

AveRecall: 0.5499999999999999

AvePrecision: 1.0

AveError: 0.45000000000000007

AveTime: 0.05553739146649517

Based on 120 training data and 40 testing data:

Default (n\_neighbors=5):

MaxRecall: 0.6

MaxPrecision: 1.0

MinError: 0.4

AveRecall: 0.5

AvePrecision: 1.0

AveError: 0.5

AveTime: 0.16307157576025738

After improve (n\_neighbors=3,algorithm='ball\_tree'):

MaxRecall: 0.575

MaxPrecision: 1.0

MinError: 0.425

AveRecall: 0.4999999999999999

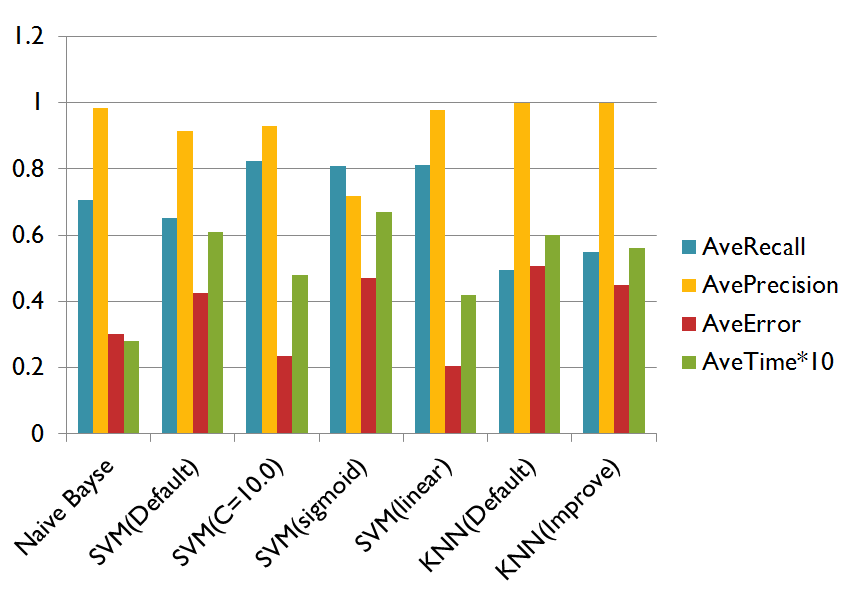
AvePrecision: 1.0

AveError: 0.5000000000000001

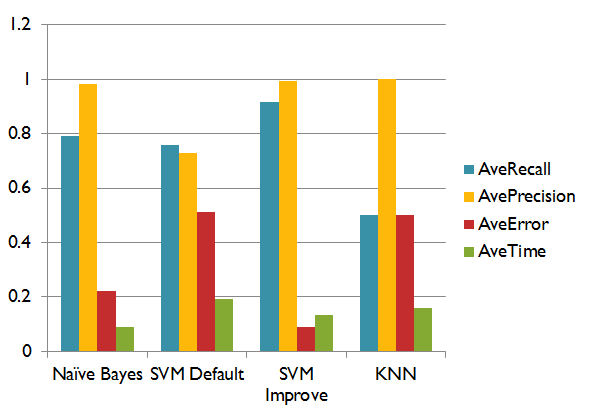
AveTime: 0.15863941423361128

# Conclusion

Each algorithm based on 60 training data and 20 testing data, there’s a chart:

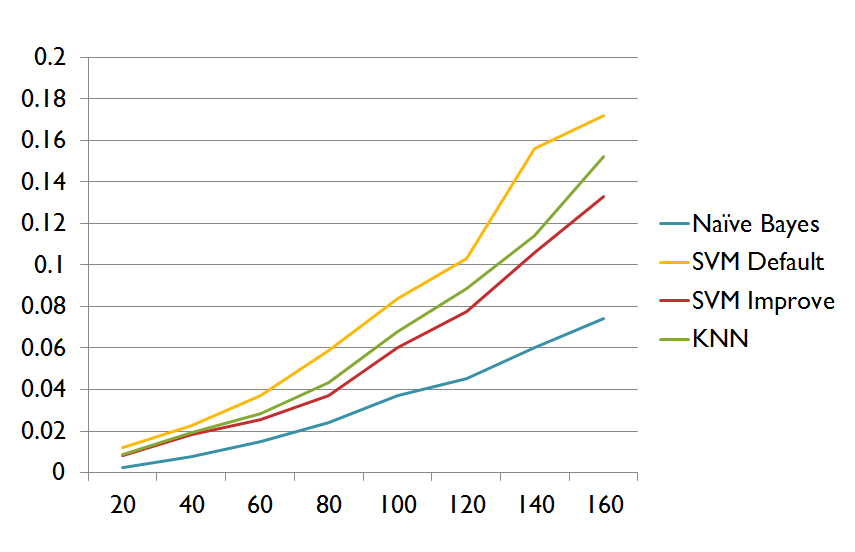


Based on 120 training data and 40 testing data:



From these data, it’s easy to find strong point and weakness about these algorithms work in SMS spam filtering system.

Naïve Bayes spend least time but have high recall and precision. SVM spend more time, but its recall and precision is better than Naïve Byes. Although KNN takes more time and recall looks not good enough, precision is always 100%. It means if you use KNN, you don’t need to worry about your important SMS filtered by the system.



This is the picture about time line of each algorithm with different data set. When increase the data set, vocabulary list grows and vector of each sentence also grows. So that, algorithm needs more space and time to calculate it. From this picture, the trend of Naive Bayes is much better than other algorithm.

# Run the Code

SMSSpamCollection.data is a Data Set, Bayesian\_mail.py is Naïve Bayes algorithm, SVM\_mail.py is SVM algorithm, KNN\_mail.py is KNN algorithm.

In addition, Bayesian\_mail.py has a function named createData(Max) that can transform SMSSpamCollection.data to the data which I can use. It means, you must run Bayesian\_mail.py first, then KNN or SVM. And I note it now, because it doesn’t need to run two times.

All my project is based on two packages, named sklearn and numpy, make sure pip install them before run the code.