MACHINE LEARNING METHODS FOR SPAM E-MAIL CLASSIFICATION

GUDUR HEMANTH KUMAR (M12504038)

KRISHNA ROHIT DATLA (M12467644)

CHALASANI SHARMILA GAYATHRI (M12510198)

SHRITI NARAPARAJU (M12057116)

PROBLEM

E-mail is an effective and very cheap means of communication, this fact is being exploited by numerous organizations for carrying out their advertisements. This process is being carried out so extensively that, e-mail inboxes of millions of people are filled with spam e-mails. This results in lot of wastage of valuable time for each user, to go through the spam e-mails. Additionally, spam emails traffic between servers results in delay in delivery of important e-mails at the right time to the users. Considering the scale and intensity of this problem, we developed a system that uses the concepts of Multinomial Naïve Bayes theorem and Supports Vector Machines to identify spam e-mails. Our spam detection system identifies emails that are spam emails and in turn saves time for millions of users.

BACKROUND READING

Naïve Bayes Classifier:

The Naive Bayes classifier is a simple probabilistic classifier which is based on Bayes theorem with strong and naïve independence assumptions. It is one of the most basic text classification techniques with various applications in email spam detection, personal email sorting, document categorization, sexually explicit content detection, language detection and sentiment detection. Naive Bayes classifier is very efficient since it is less computationally intensive (in both CPU and memory) and it requires a small amount of training data. Moreover, the training time with Naive Bayes is significantly smaller as opposed to alternative methods.

Support Vector Machines:

Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression

analysis. They have High accuracy, good theoretical guarantees regarding overfitting, and with an appropriate kernel they can work well even if data is not linearly separable in the base feature space. Especially popular in text classification problems where very high-dimensional spaces are the norm.

Term Frequency-Inverse Document Frequency(TFIDF):

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, but is often 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. Nowadays, tf-idf is one of the most popular term-weighting schemes; 83% of text-based recommender systems in the domain of digital libraries use tf-idf

Gaussian Naïve Bayes:

It is used in classification and it assumes that features follow a normal distribution.

Multinomial Naïve Bayes:

It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".

Bernoulli Naïve Bayes:

The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words'

model where the 1s & 0s are “word occurs in the document” and “word does not occur in the document” respectively.

OUR APPROACH

Data Set Description

We have put together the data set with spam data collected from Kaggle website and UCI repository. The dataset contains 5573 instances with 5573 rows and 2 columns labelled as ‘Category’ and ‘Message’ respectively. Each message or mail is classified as ham (not spam) or spam. Source dataset is raw and is not pre-processed.

Preprocessing

We implemented the concept of TFIDF(term frequency-inverse document frequency) in order to remove words that are common in e-mails irrespective of them being spam or not spam. We also eliminated all numeric and alpha numeric values from the source dataset. We identified all the stop words (ex:a, an, the) by identifying the words with abnormally high frequencies and removed them, because they do not play a role in determining whether an e-mail is spam or not.

Machine Learning Techniques

We considered 80 percent of data for training and 20 percent of data as test data and for implementing “Multinomial Naïve Baye’s Classifier” and “Support Vector Machines”. We installed dependencies from SKLearn package to implement them. In Naïve Bayes classifier for text classification we can either use the concepts of Gaussian Naïve Bayes or Multi-variate Bernoulli Naïve Baye’s or Multinomial Naïve baye’s. In our case, we have implemented Multinomial Naïve Bayes because in spam detection, frequency of a word also plays a role in determining if an e-mail is spam or not and Multinomial Naïve Bayes does exactly that. In addition to

Multinomial Naïve Bayes Classifier we also implemented Support Vector Machines with the same dataset. For evaluating which of these is a better classifier for this problem we created the confusion matrix and found out the accuracy and F-score of both the cases.

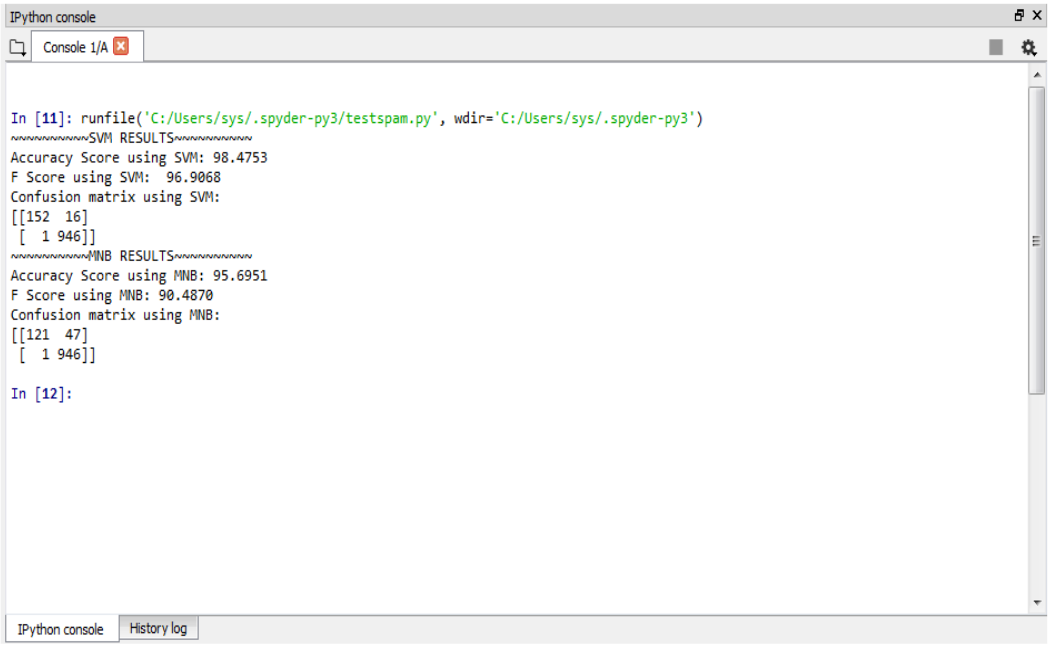
CODE

```
# -*- coding: utf-8 -*-
"""
Created on Sat Dec  2 02:21:51 2017

@author: sys
"""

# import all dependencies
# -*- coding: utf-8 -*-
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score

# read the data and replace null values with a null string
df1 = pd.read_csv("spamham.csv")
df = df1.where((pd.notnull(df1)), '')
```

```
# Categorize Spam as 0 and Not spam as 1

df.loc[df["Category"] == 'ham', "Category",] = 1

df.loc[df["Category"] == 'spam', "Category",] = 0

# split data as label and text . System should be capable of predicting the label based on the text

df_x = df['Message']

df_y = df['Category']

# split the table - 80 percent for training and 20 percent for test size

x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, train_size=0.8, test_size=0.2, random_state=4)


# feature extraction, coversion to lower case and removal of stop words using TFIDF VECTORIZER

tfvec = TfidfVectorizer(min_df=1, stop_words='english', lowercase=True)

x_trainFeat = tfvec.fit_transform(x_train)

x_testFeat = tfvec.transform(x_test)


# SVM is used to model

y_trainSvm = y_train.astype('int')

classifierModel = LinearSVC()

classifierModel.fit(x_trainFeat, y_trainSvm)

predResult = classifierModel.predict(x_testFeat)


# GNB is used to model

y_trainGnb = y_train.astype('int')

classifierModel2 = MultinomialNB()

classifierModel2.fit(x_trainFeat, y_trainGnb)

predResult2 = classifierModel2.predict(x_testFeat)
```

```
# Calc accuracy,converting to int - solves - cant handle mix of unknown and binary

y_test = y_test.astype('int')

actual_Y = y_test.as_matrix()


print("~~~~~~~~~~SVM RESULTS~~~~~~~~~~")

#Accuracy score using SVM

print("Accuracy Score using SVM: {0:.4f}".format(accuracy_score(actual_Y, predResult)*100))

#FScore MACRO using SVM

print("F Score using SVM: {0: .4f}".format(f1_score(actual_Y, predResult,
average='macro')*100))

cmSVM=confusion_matrix(actual_Y, predResult)

#"[True negative  False Positive\nFalse Negative True Positive]"

print("Confusion matrix using SVM:")

print(cmSVM)

print("~~~~~~~~~~MNB RESULTS~~~~~~~~~~")

#Accuracy score using MNB

print("Accuracy Score using MNB: {0:.4f}".format(accuracy_score(actual_Y, predResult2)*100))

#FScore MACRO using MNB

print("F Score using MNB:{0: .4f}".format(f1_score(actual_Y, predResult2,
average='macro')*100))

cmMNb=confusion_matrix(actual_Y, predResult2)

#"[True negative  False Positive\nFalse Negative True Positive]"

print("Confusion matrix using MNB:")

print(cmMNb)
```

RESULTS COMPARISION AND CONCLUSION



It is clear from the results that Support vector machines outperforms the multinomial NBC in detection of spam mails. Even though it is a small difference and multinomial NBC also does a decent job we have to always build the better machine to solve our problems. Hence, SVM is better at filtering spam mails from non-spam mails.

	ACCURACY	F SCORE
SUPPORT VECTOR MACHINE	98.4753	96.9068
MULTINOMIAL NBC	95.6951	90.4870

REFERENCES:

1) *https://www.wikipedia.org*
2) *www.kaggle.com*
3) *www.enggjournals.com/ijet/docs/IJET17-09-02-310.pdf*
4) *https://archive.ics.uci.edu/ml/datasets/spamdataset*
5) *http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/*
6) *International Journal of Computer Science & Information Technology (IJCSIT),Vol 3, No 1, Feb 2011 DOI : 10.5121/ijcsit.2011.3112 173 "*MACHINE LEARNINGMETHODS FOR SPAM E-MAIL CLASSIFICATION*".*
7) *"*COMPARISON AND ANALYSIS OF SPAM DETECTION ALGORITHM*" Sahil Puri1, Dishant Gosain2, Mehak Ahuja3, Ishita Kathuria4, Nishtha Jatana5*