Configuration Manual

Ensemble Classification for Email spam Prediction

Msc Cyber Security

Chinedu Udogwu

Student ID: 19222360

School of Computing

National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Chinedu Timothy Udogwu
Student ID:	19222360
Programme:	Cyber Security **Year:** 2021
Module:	Msc Internship
Lecturer:	Imran Khan
Submission Due Date:	16/12/2021
Project Title:	Ensemble classification method for spam email prediction
Word Count:	500 **Page Count:** 25

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Chinedu Timothy Udogwu

Date: ……………….16/12/2021……………………………………………………………………… ……………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Chinedu Udogwu
Student ID: 19222360

1. Introduction

This Configuration Manual explains how to complete the thesis topic "Ensemble categorization for spam prediction" step by step. The major goal of this research was to show that in the context of Machine Learning and Monitoring, a mixture of ML algorithms performs better than a single algorithm (cyber security). The software and hardware requirements for this study are detailed in the manual. It also includes instructions on how to set up some of the instruments that were utilized throughout the research. In the following sections, we'll go over the libraries and packages we used in our research.

2. System Requirement

2.1 Hard Ware Requirement

As this study was conducted on the local system, the local system had the following specifications:

1. System OS – Windows 10 Home 64-bit
2. Processor – Intel Core i5 8th Gen @ 1.80 GHz
3. Ram – 12 GB DDR4 @ 2400 MHz
4. Hard Disk Drive – 1 TB 5400 rpm SATA

3.1 Software requirement

1. Python 3.8 – This is a programming language that was used throughout this study from pre-processing the data to evaluating the machine learning models for the two approaches.
2. Jupyter Notebook – This is an open-source web application that helps to code and execute Python and also helps to visualize the data [1]. For this study, the Jupyter Notebook 6.0.3 was used to execute Python 3.8.
3. Node.js 12.18.2 – This is an open-source, cross-platform tool that is used to run a JavaScript code outside a browser [2]. This tool is used to run the tool Plato for this study.
4. Google colab - Alternative to the jupyter notebook environment, was the Google-Colab platform, which comes with pre-installed basic libraries for machine learning applications.
5. Microsoft Excel – This tool is used to import the datasets. All the datasets used for this study is used in .csv file format. This tool was also used to visualise the results of the evaluation of the different models.

Importing the library

In [63]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing the dataset

In [64]:

```
df = pd.read_csv('../input/spam-dataset/spam.csv')
```

In [65]:

```
df.head()
```

Out[65]:

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

Dataset preprocessing

In [66]:

```
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [67]:

```
df.groupby('label').describe()
```

Out[67]:

Graphical Representation

In [72]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [73]:

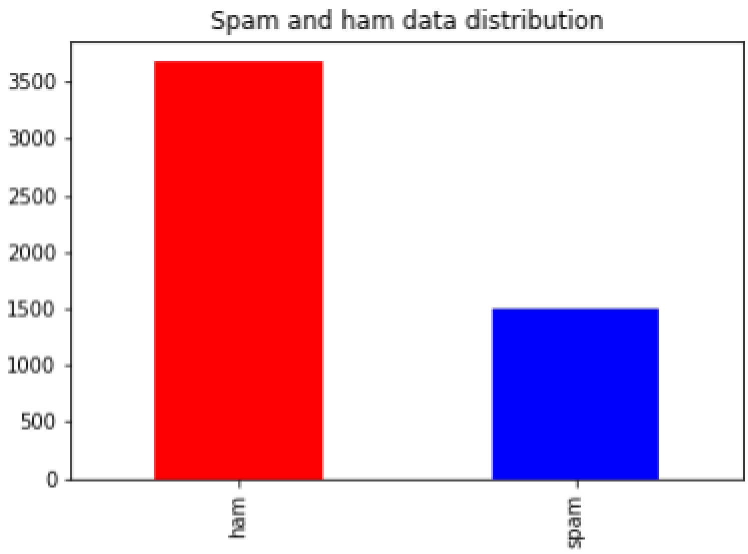
```
df['length'].plot(bins=50,kind='hist')
```

Out[73]:

```
<AxesSubplot:ylabel='Frequency'>
```

In [77]:

```python
#Visualizing Data
count_Class=pd.value_counts(df["label"], sort= True)
count_Class.plot(kind= 'bar', color= ["red", "blue"])
plt.title('Spam and ham data distribution')
plt.show()
```



Text Pre processing

In [85]:

```python
import string
mess = 'sample message!...'
nopunc=[char for char in mess if char not in string.punctuation]
nopunc=''.join(nopunc)
print(nopunc)
```

```
sample message
```

In [86]:

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords.words('english')[0:10]
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_dat
a...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[86]:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'yo
u', "you're"]
```

In [87]:

```
def text_process(mess):
    nopunc =[char for char in mess if char not in string.punctuation]
    nopunc=''.join(nopunc)
    return [word for word in nopunc.split() if word.lower() not in stopwords.wor
ds('english')]
```

Converting text to

In [91]:

```
import re
def preprocess(text):
    text = text.replace('\r',' ')
    text = text.replace('\n',' ')
    text = text.replace('#',' ')
    text = text.replace("we ' re","we are")
    text = text.replace("they ' re","they are")
    text = text.replace("you ' re","you are")
    text = text.replace("Subject:"," ")

    return text
```

In [92]:

```
df['text_clean']=df['text'].map(preprocess)
```

Converting text_clean to vector

In [94]:

```
from sklearn.feature_extraction.text import CountVectorizer


count_vectorizer = CountVectorizer(ngram_range=(1,1))
X = count_vectorizer.fit_transform(df['text_clean'])
print(X.shape)
```

```
(5171, 50447)
```

Split dataset into the training and testing samples

In [95]:

```
from sklearn.model_selection import train_test_split

x,x_test,y,y_test = train_test_split(X,df['label_num'],test_size=0.2,random_stat
e=42)
x_train,x_cv,y_train,y_cv = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [96]:

```
print((x_train).shape)
```

```
(3308, 50447)
```

Naive bayes classifier

model on training sample

In [97]:

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from time import time
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
start = time()
gnb.fit(x.toarray(),y)
end = time()
train_time_nb = end-start
y1_pred = gnb.predict(x.toarray())
print(classification_report(y, y1_pred))
print(confusion_matrix(y, y1_pred))
print(accuracy_score(y, y1_pred))
print("Train time for Naive Bayes",train_time_nb,"s")
```

Model on testing samples

In [98]:

```
start = time()
y_pred_gnb = gnb.predict(x_test.toarray())
end = time()
test_time_nb = end-start
# printing the test time
print("Test time for Naive Bayes:", test_time_nb)
print(classification_report(y_test, y_pred_gnb))
print(confusion_matrix(y_test, y_pred_gnb))
print(accuracy_score(y_test, y_pred_gnb))
```

```
Test time for Naive Bayes: 0.9663674831390381
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       742
           1       0.96      0.87      0.92       293

    accuracy                           0.95      1035
   macro avg       0.96      0.93      0.94      1035
weighted avg       0.95      0.95      0.95      1035

[[732  10]
 [ 37 256]]
```

support vector classifier (SVC)

model on training sample

In [99]:

```
from sklearn.svm import SVC
svc=SVC( kernel='linear')
start = time()
svc.fit(x.toarray(),y)
end = time()
train_time_svc = end-start
# printing the train time
print("Train time for SVM:", train_time_svc)
y3_pred =svc.predict(x.toarray())
print(classification_report(y, y3_pred))
print(confusion_matrix(y, y3_pred))
print(accuracy_score(y, y3_pred))
```

```
Train time for SVM: 214.95182394981384
```

Model on testing samples

In [112]:

```
start = time()
y_pred_svc = svc.predict(x_test.toarray())
end = time()
test_time_svc = end-start
# printing the test time
print("Test time for SVM:", test_time_svc)
print(classification_report(y_test, y_pred_svc))
print(confusion_matrix(y_test, y_pred_svc))
print(accuracy_score(y_test, y_pred_svc))
```

```
Test time for SVM: 49.07707715034485
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       742
           1       0.94      0.94      0.94       293

    accuracy                           0.97      1035
   macro avg       0.96      0.96      0.96      1035
weighted avg       0.97      0.97      0.97      1035

[[725  17]
 [ 19 274]]
0.9652173913043478
```

Decision Tree (DT)

model on training sample

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
start = time()
classifier.fit(x.toarray(),y)
end = time()
train_time_dt = end-start
# printing the train time
print("Train time for Decision Tree:", train_time_dt)
y5_pred =classifier.predict(x.toarray())
print(classification_report(y, y5_pred))
print(confusion_matrix(y, y5_pred))
print(accuracy_score(y, y5_pred))
```

Model on testing samples

In [102]:

```
start = time()
y_pred_dt = classifier.predict(x_test.toarray())
end = time()
test_time_dt = end-start
# printing the test time
print("Test time for Decision Tree:", test_time_dt)
print(classification_report(y_test, y_pred_dt))
print(confusion_matrix(y_test, y_pred_dt))
print(accuracy_score(y_test, y_pred_dt))
```

Vooting Classifier

In [103]:

```
# using mlxtend's ensemble voting classifier to use the pretrained models
from mlxtend.classifier import EnsembleVoteClassifier
```

In [104]:

```
vot_clf = EnsembleVoteClassifier(clfs=[gnb,svc,classifier],weights=[1,1,1],fit_b
ase_estimators=False,voting="hard")
start = time()
vot_clf = vot_clf.fit(x.toarray(),y)
end = time()
train_time_vot = end-start
print("Train time for Voting Classifier with pretrained models:", train_time_vot
)
y_pred = vot_clf.predict(x.toarray())
print(classification_report(y, y_pred))
print(confusion_matrix(y, y_pred))
print(accuracy_score(y, y_pred))
```

In [105]:

```
start = time()
y_pred_vote = vot_clf.predict(x_test.toarray())
end = time()
test_time_vot = end-start
print("Test time for Voting Classifier:", test_time_vot)
score = accuracy_score(np.array(y_test), y_pred_vote)
print("Voting Classifier Score % f" % score)
```

```
Test time for Voting Classifier: 49.95113658905029
Voting Classifier Score  0.979710
```

Plotting Performance of the models

In [106]:

```
from sklearn.metrics import precision_score,recall_score,f1_score
```

F1 Score

In [107]:

```
classes=["NB","SVM","DT","Vote"]
f1_scores = [f1_score(y_test, y_pred_gnb),f1_score(y_test, y_pred_svc),f1_score(
y_test, y_pred_dt),f1_score(y_test, y_pred_vote)]
Class_index=np.arange(len(classes))
plt.bar(Class_index,f1_scores,color="y",linestyle="--",edgecolor="r")
plt.xticks(Class_index+width,classes)
plt.title("F1 Score")
plt.xlabel("Models")
plt.ylabel("Ratio ")
plt.legend()
plt.show()
```

In [108]:

```
classes=["NB","SVM","DT","Vote"]
# accuracy of the models
accuracy = [accuracy_score(y_test, y_pred_gnb),accuracy_score(y_test, y_pred_svc
),accuracy_score(y_test, y_pred_dt),accuracy_score(y_test, y_pred_vote)]
Class_index=np.arange(len(classes))
plt.bar(Class_index,accuracy,color="r",linestyle="--",edgecolor="r",align="edge"
)
plt.xticks(Class_index+width,classes)
plt.title("Accuracy  ")
plt.xlabel("Models")
plt.ylabel("percentage ")
plt.legend()
plt.show()
```

References

[1] Maria Habib, Hossam Faris, Mohammad A. Hassonah, Ja'farAlqatawna, Alaa F. Sheta, Ala' M. Al-Zoubi, “Automatic Email Spam Detection using Genetic Programming with SMOTE”, 2018, Fifth HCT Information Technology Trends (ITT)

[2] Amany A. Naem, Neveen I. Ghali, Afaf A. Saleh, “Antlion optimization and boosting classifier for spam email detection”, 2018, Future Computing and Informatics Journal