Configuration Manual

Academic Internship
MSc Cybersecurity

Suman Verma
Student ID: x21154996

School of Computing
National College of Ireland

Supervisor: Dr. Vanessa Ayala Rivera

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Suman Verma
Student ID:	x21154996
Programme:	MSc Cybersecurity **Year:** 2022
Module:	Academic Internship
Lecturer:	…………………………………………………………………………………
Submission Due Date:	15/12/2022
Project Title:	Detection of Phishing in Mobile Instant Messaging using Natural Language Processing and Machine Learning
Word Count:	**1023** **Page Count: 09**

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: Suman verma

Date: 15/12/2022

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

Suman Verma
x21154996

1 Introduction

The configuration manual is a methodical procedure for the project ‘Detection of Phishing in Instant Message using Natural Language Processing and Machine Learning’. It involves the technical steps, Installations, and Implementation for the purpose of project mentioned above. The objective to present this configuration is to guide and assist readers through each step of the process so they can produce the desired results and output, which are delivered in a technical report.

1.1 Project Overview

The project is aimed towards detection of phishing in Instant Message as proactive measures to safeguard users from social engineering-based phishing attack. To achieve the goal, we have aimed to use NLP (Natural Language Processing) to pre-process and extract the features of phishing present in the text of message and use these to train different machine learning models to help them classify the Instant messages as phishing and no phishing. The proposed model if applied with other factors can help to detect phishing in the instant message application and can thus protect the users and save the business.

2 Hardware/ software requirements:

2.1 Hardware:

Processor: Intel i5
Memory: 16GB
Operating System: Windows 11, 64-bit

2.2 Software:

Jupyter -Lab: Python -programming language software with Jupyter-lab from anaconda distribution has been installed for the purpose of using NLP and developing machine learning models.

Draw.io has been used to create the flowchart of proposed model and framework.
Microsoft Excel used for presentation of result in chart/graph.

2.3 Dataset:

The “SMS phishing dataset for machine learning and pattern recognition” used for the project was open source and available on Mendeley data contributed by Mishra S. and Soni D. (2022). It had a collection of labelled text messages used in SMS phishing research. It contained 5971 texts messages labelled as Legitimate /Ham (4844), Spam (489), and Smishing (638).

3 Implementation of project in Jupyter

3.1 Pre-processing of dataset

Data cleaning and pre-processing has been done in Jupyter environment using NLP. Fig 1 shows code for importing different libraries.

```
In [1]: #Importing all libraries

        import pandas as pd
        import numpy as np

        import nltk
        #nltk.download ()

        from nltk.corpus import stopwords
        from nltk.tokenize import word_tokenize
        from nltk.stem.wordnet import WordNetLemmatizer

        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.naive_bayes import GaussianNB
        from sklearn import metrics
        from collections import Counter

        from sklearn import preprocessing
        import matplotlib.pyplot as plt
        plt.rc("font", size=14)

In [2]: #importing dataset
        data =pd.read_csv(r"C:\Users\suman\OneDrive\Desktop\Project thesis\SMS dataset.csv")
```

Fig 1 Importing libraries and Dataset in the Jupyter environment

Pre-processing of dataset by converting to lower case and removal of special charcter,numeric charater has been shown in Fig 2 and Fig 3 shows removal of shortword, stopword and lemmatization by word tokenization.

```
In [10]: #convert text to lower case
         textdata_lower = textdata_phone.str.lower()
         textdata_lower.head()

Out[10]: 0    your opinion about me? 1. over 2. jada 3. kusr...
         1    what's up? do you want me to come online? if y...
         2                         so u workin overtime nigpun?
         3    also sir, i sent you an email about how to log...
         4    please stay at home. to encourage the notion o...
         Name: TEXT, dtype: object

In [11]: #Remove special characters
         textdata_spechar = textdata_lower.apply(lambda x: re.sub(r'[^\w\s]', "", x))
         textdata_spechar.head()

Out[11]: 0    your opinion about me 1 over 2 jada 3 kusruthi...
         1    whats up do you want me to come online if you ...
         2                          so u workin overtime nigpun
         3    also sir i sent you an email about how to log ...
         4    please stay at home to encourage the notion of...
         Name: TEXT, dtype: object

In [12]: #removing numeric character
         import re
         pattern = r'[0-9]'
         textdata_nonumeric = textdata_spechar.apply(lambda x: re.sub(pattern,"", x))
         textdata_nonumeric[0]
```

Fig 2 Pre-processing of data by converting to lower case and removal of special charcter,numeric charater

```
In [13]: #removing shortword
         shortword = re.compile(r'\W*\b\w{1,3}\b')
         textdata_noshortword = textdata_nonumeric.apply(lambda x: shortword.sub('', x))
         textdata_noshortword[3]

Out[13]: 'also sent email about into payment portal send another message that should explain things back home have great weekend'

In [14]: #Remove stopwords
         textdata_stopword = textdata_noshortword.apply(lambda x: ' '.join([word for word in word_tokenize(x) if not word in set(stopwords
         textdata_stopword[228]

Out[14]: 'dear xxxxxxx invited xchat final attempt contact chat pmsgrcvdhgsuitelandsrowwjhl'

In [15]: #Lemmatize Text
         textdata_lemmatized = textdata_stopword.apply(lambda x: ' '.join([WordNetLemmatizer().lemmatize(w) for w in word_tokenize(x)]))
         textdata_lemmatized.head()

Out[15]: 0    opinion jada kusruthi lovable silent character...
         1             whats want come online free talk sometime
         2                           workin overtime nigpun
         3    also sent email payment portal send another me...
         4    please stay home encourage notion staying home...
         Name: TEXT, dtype: object
```

Fig 3 Pre-processing by removal of shortword, stopword and lemmatization by word tokenization

Data preprocessing using Gensim library in case of Classification with Word2vec method of vectorization has been shown in Fig 4.

```
# Clean data using the built in cleaner in gensim
data['text_clean'] = data['TEXT'].apply(lambda x: gensim.utils.simple_preprocess(x))
data.head()
```

Fig 4 Data preprocessing using Gensim library

3.2 Vectorization

3.2.1 Vectorization using Bag of Words

Fig 5 shows vectorization using Bag of Words with unigram and bigram depending upon the frequency of the words in dataset.

```
In [28]: # Fit the CountVectorizer to the training data specifiying a minimum
         # document frequency of 5 and extracting 1-grams and 2-grams
         vect = CountVectorizer(min_df=5, ngram_range=(1,2)).fit(textdata_lemmatized.tolist())

         X_train_vectorized = vect.transform(textdata_lemmatized.tolist())
```

Fig 5 Vectorization using Bag of Words

3.2.2 Vectorization using TFIDF

Fig 6 shows Vectorization using TFIDF. TF-IDF helps to create vector using TF(Term frequency) and IDF(Inverse document frequency) that gives more weight to the less frequently but important words in dataset.

```
In [21]: #TfidVectorizer
         from sklearn.feature_extraction.text import TfidfVectorizer
         vect = TfidfVectorizer()
         transformed_output =vect.fit_transform(textdata_lemmatized)
```

Fig 6 Vectorization using TFIDF

3.2.3 Vectorization using Word2vec

Fig 7 shows Vectorization using word2vec. Word2vec is dense representation of text with similar representation of similar words in dataset showing contextual and semantic relationship. Gensim library has been used for the purpose of Word2vec word embedding.

```
In [46]: # Train the word2vec model
         w2v_model = gensim.models.Word2Vec(X_train,
                                            vector_size=100,
                                            window=5,
                                            min_count=2)
```

```
In [47]: words = set(w2v_model.wv.index_to_key )
         X_train_vect = np.array([np.array([w2v_model.wv[i] for i in ls if i in words])
                                  for ls in X_train], dtype=object)
         X_test_vect = np.array([np.array([w2v_model.wv[i] for i in ls if i in words])
                                  for ls in X_test], dtype=object)
```

Fig 7 shows Vectorization using word2vec

3.3 Train Test split

Vectorized data has been split into train and test set with a test size of 0.33. Fig 8 shows train and test split in case of BOW and TFIDF. Fig 9 shows train and test split with Word2vec.

```
In [31]: #split the data into test and train set
         X_train, X_test, y_train, y_test = train_test_split(clean_data.iloc[:, 1:], clean_data['PHISHING'] , test_size=0.33, random_state
```

Fig 8 Train and test split with BOW of vectorisation

```
In [42]: from imblearn.over_sampling import RandomOverSampler

In [43]: X_train, X_test, y_train, y_test = train_test_split(final_data['TEXT'], final_data['PHISHING'] , test_size=0.33, random_state=42)
```

```
In [52]: ROS = RandomOverSampler(sampling_strategy=1)
```

```
In [54]:
         X_train_ros, y_train_ros = ROS.fit_resample(X_train_tf, y_train)
```

Fig 9 Train and test split with TFIDF method of vectorisation and Rnadom Over Sampling

```
In [9]: # Split data into train and test sets
        X_train, X_test, y_train, y_test = train_test_split (data['text_clean'], data['LABEL'] , test_size=0.33)
```

Fig 10 Train and test split with Word2vec method of vectorisation

3.4 Applying Machine Learning Models

3.4.1 BOW vector with Logical Regression

```
In [36]: ## Apply Logisitic Regression Algorithm to classify the data
         from sklearn.linear_model import LogisticRegression
         from sklearn import metrics

         logreg = LogisticRegression()
         logreg.fit(X_train, y_train)
```

Fig 11 Applying Logistic Regression on BOW vector

3.4.2 BOW vector with Gaussian Naïve Bayes

```
In [66]: #Create a Gaussian Classifier
         gnb = GaussianNB()
         # Train the model using the training sets
         gnb.fit(X_train, y_train)
```

Fig 12 Applying Gaussian Naïve Bayes on BOW vector

3.4.3 BOW vector with Random Forest

```
In [75]: #RandomForestClassifier
         from sklearn.ensemble import RandomForestClassifier
         rf = RandomForestClassifier(n_estimators=40)
         rf.fit(X_train, y_train)
```

Fig 13 Applying Random Forest on BOW vector

3.4.4 TFIDF vector with Logical Regression

```
In [50]: ## Apply Logisitic Regression Algorithm to classify the data on balanced data
         from sklearn.linear_model import LogisticRegression
         from sklearn import metrics

         logreg_balanced = LogisticRegression()
         logreg_balanced.fit(X_train_ros, y_train_ros)
```

Fig 14 Applying Logistic Regression on TFIDF vector

3.4.5 TFIDF vector with Gaussian Naïve Bayes

```
In [56]: #Create a Gaussian Classifier
         from sklearn.naive_bayes import GaussianNB
         gnb_balanced = GaussianNB()
         # Train the model using the training sets
         gnb_balanced.fit(X_train_ros, y_train_ros)
```

Fig 15 Applying Gaussian Naïve Bayes on TFIDF vector

3.4.6 TFIDF vector with Random Forest

```
In [82]: # Instantiate and fit a basic Random Forest model on top of the vectors
         from sklearn.ensemble import RandomForestClassifier
         rf = RandomForestClassifier()
         rf.fit(X_train_ros, y_train_ros)
```

Fig 16 Applying Random Forest on TFIDF vector

3.4.7 Word2vec with Random Forest

```
In [21]: # Instantiate and fit a basic Random Forest model on top of the vectors
         from sklearn.ensemble import RandomForestClassifier
         rf = RandomForestClassifier()
         rf_model = rf.fit(X_train_vect_avg, y_train.values.ravel())
```

Fig 17 Applying Random Forest on Word2vec

3.5 Evaluation

3.5.1 Evaluation of Classifiers on BOW vector

- Evaluation of Logical Regression

The models are evaluated on the accuracy percentage for classification and recall percentage of phishing messages.

```
y_pred_log = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))

Accuracy of logistic regression classifier on test set: 0.97

from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(y_test, y_pred_log)
print(confusion_matrix)

[[1742   17]
 [  45  167]]

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_log))

              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1759
           1       0.91      0.79      0.84       212

    accuracy                           0.97      1971
   macro avg       0.94      0.89      0.91      1971
weighted avg       0.97      0.97      0.97      1971
```

Fig 18 Evaluating Logical Regression with BOW

- Evaluation of Gaussian Naïve Bayes Classifier

```
#Predict Output
y_pred_gnb = gnb.predict(X_test)

#Import scikit-learn metrics module for accuracy calculation
#from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_gnb))

Accuracy: 0.8356164383561644
```

Fig 19 Evaluating Gaussian Naïve Bayes with BOW

- Evaluation of Random Forest Classifier

```
In [50]: #RandomForestClassifier
         from sklearn.ensemble import RandomForestClassifier
         rf = RandomForestClassifier(n_estimators=40)
         rf.fit(X_train, y_train)
         y_pred_rf = rf.predict(X_test)
         rf.score(X_test, y_test)

Out[50]: 0.9700659563673262

In [51]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred_rf))

                       precision    recall  f1-score   support

                    0       0.97      0.99      0.98      1759
                    1       0.92      0.79      0.85       212

             accuracy                           0.97      1971
            macro avg       0.95      0.89      0.92      1971
         weighted avg       0.97      0.97      0.97      1971
```

Fig 20 Evaluating Random Forest with BOW

3.5.2 Cross validation of models with BOW method of vectorization

Cross validation avoids over fitting of data on machine learning models and helps to evaluate without any bias. Models were cross-validated using stratified K-Fold cross validation method, where k=10.

```
In [52]: #KFold Cross validation
         #Logistic regression model performance using cross_val_score
         from sklearn.model_selection import cross_val_score
         cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'), clean_data.iloc[:, 1:], clean_data['PHISHING'],cv=10)

Out[52]: array([0.96655518, 0.97319933, 0.97152429, 0.98492462, 0.9681742 ,
                0.96482412, 0.96984925, 0.96314908, 0.97487437, 0.97487437])

In [53]: #random forest performance using cross_val_score
         cross_val_score(RandomForestClassifier(n_estimators=40), clean_data.iloc[:, 1:], clean_data['PHISHING'],cv=10)

Out[53]: array([0.95317726, 0.96984925, 0.97654941, 0.98659966, 0.9681742 ,
                0.96482412, 0.96482412, 0.96482412, 0.97654941, 0.97822446])

In [54]: ##GaussianNB performance using cross_val_score
         cross_val_score(GaussianNB(), clean_data.iloc[:, 1:], clean_data['PHISHING'],cv=10)

Out[54]: array([0.80769231, 0.80234506, 0.81574539, 0.82077052, 0.82747069,
                0.81407035, 0.8040201 , 0.8040201 , 0.79731993, 0.81072027])
```

Fig 21 Cross validation of models with BOW

3.5.3 Parameter tuning of Random Forest Model

n_estimator that is no. of trees in forest as per ML model was tuned, and the best performance was obtained when n_estimator=40.

```
In [58]: scores4 = cross_val_score(RandomForestClassifier(n_estimators=40),clean_data.iloc[:, 1:], clean_data['PHISHING'], cv=10)
         np.average(scores4)

Out[58]: 0.9706934897452705
```

Fig 22 Parameter tuning of Random Forest with BOW

3.5.4 Evaluation of Classifiers on TFIDF vector

- Evaluation of Logical Regression

```
In [51]: y_pred_log_balanced = logreg_balanced.predict(X_test_tf)
         print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg_balanced.score(X_test_tf, y_test)))
```

- Evaluation of Gaussian Classifier

```
In [57]: y_pred_gnb_balanced = gnb_balanced.predict(X_test_tf)
         print('Accuracy of Naive Bayes classifier on test set: {:.2f}'.format(gnb_balanced.score(X_test_tf, y_test)))
```

- Evaluation of Random Forest Classifier

```
In [83]: # Use the trained model to make predictions on the test data
         y_pred_rf = rf.predict(X_test_tf)
```

Fig 23 ML Classifiers with Tf-IDF

3.5.5 Cross validation of models with TF-IDF method of vectorization

```
In [60]: #KFold Cross validation
         #Logistic regression model performance using cross_val_score
         from sklearn.model_selection import cross_val_score
         cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'), X_train_ros, y_train_ros,cv=10)

Out[60]: array([0.98741259, 0.99300699, 0.97762238, 0.97622378, 0.98321678,
                0.97482517, 0.98461538, 0.99160839, 0.97478992, 0.97478992])

In [61]: ##GaussianNB performance using cross_val_score
         cross_val_score(GaussianNB(), X_train_ros, y_train_ros,cv=10)

Out[61]: array([0.94125874, 0.93706294, 0.93706294, 0.93006993, 0.94685315,
                0.92307692, 0.92587413, 0.93846154, 0.94677871, 0.92577031])
```

Fig 24 Cross validation of models with TF-IDF

3.5.6 Parameter tuning of Random Forest Model

n_estimator was tuned, and the best performance was obtained when n_estimator=40.

```
In [66]: scores4 = cross_val_score(RandomForestClassifier(n_estimators=40),X_train_ros, y_train_ros, cv=10)
         np.average(scores4)

Out[66]: 0.9928649781590957
```

Fig 25 Parameter tuning of Random Forest with BOW

3.5.7 Evaluation of Random Forest Classifier Classifiers on Word2vec method of vectorization

```
In [22]: # Use the trained model to make predictions on the test data
         y_pred_rf = rf_model.predict(X_test_vect_avg)
```

Fig 26 Evaluating Random Forest with Word2vec

3.5.8 Cross validation of Random Forest Classifier with Word2vec vector

```
In [36]: #random forest performance using cross_val_score
         from sklearn.model_selection import cross_val_score
         cross_val_score(RandomForestClassifier(n_estimators=40), X_train_vect_avg, y_train.values.ravel(),cv=10)

Out[36]: array([0.94  , 0.9575, 0.94  , 0.9475, 0.9425, 0.9475, 0.96  , 0.95  ,
                0.9475, 0.9625])
```

Fig 27 Cross validation of Random Forest with Word2vec

3.5.9 Parameter tuning of Random Forest Model

n_estimator was tuned, and the best performance was obtained when n_estimator=20.

```
In [38]: scores2 = cross_val_score(RandomForestClassifier(n_estimators=20),X_train_vect_avg, y_train.values.ravel(), cv=10)
         np.average(scores2)

Out[38]: 0.952
```

Fig 28 Parameter tuning of Random Forest with Word2vec

References

1. *Naive Bayes classifiers* (2022) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/naive-bayes-classifiers/ (Accessed: November 14, 2022).
2. Mishra, Sandhya; Soni, Devpriya (2022), "SMS PHISHING DATASET FOR MACHINE LEARNING AND PATTERN RECOGNITION", Mendeley Data, V1, doi: 10.17632/f45bkkt8pr.1
3. S, R.A. (2022) *How to convert list to string in python? here are the 6 ways to know: Simplilearn, Simplilearn.com*. Simplilearn. Available at: https://www.simplilearn.com/tutorials/python-tutorial/list-to-string-in-python (Accessed: October 14, 2022).
4. Splunktool (no date) *Removing alphanumeric words, with some exceptions in python 3*, *splunktool*. Available at: https://splunktool.com/removing-alphanumeric-words-with-some-exceptions-in-python-3 (Accessed: October 21, 2022).
5. Li, S. (2019) *Building a logistic regression in Python, step by step*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8 (Accessed: November 24, 2022).
6. *1. supervised learning* (no date) *scikit*. Available at: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning (Accessed: September 25, 2022).
7. *Merge, join, Concatenate and compare#* (no date) *Merge, join, concatenate and compare - pandas 1.5.2 documentation*. Available at: https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html (Accessed: October 5, 2022).
8. *Remove numbers from string in Python* (no date) *Studytonight.com*. Available at: https://www.studytonight.com/python-howtos/remove-numbers-from-string-in-python (Accessed: November 24, 2022).
9. Tripadvisor, J.-C.C.S.E.O.S.at (no date) *How to use classification report in Scikit-Learn (python), JC Chouinard*. Available at: https://www.jcchouinard.com/classification-report-in-scikit-learn/ (Accessed: December 1, 2022).
10. Brownlee, J. (2020) *Tune hyperparameters for Classification Machine Learning Algorithms*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/ (Accessed: November 23, 2022).
11. *Generating word cloud in python* (2021) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/generating-word-cloud-python/ (Accessed: November 14, 2022).
12. Codebasics (no date) *PY/12_k_fold.ipynb at master · codebasics/py*, *GitHub*. Available at: https://github.com/codebasics/py/blob/master/ML/12_KFold_Cross_Validation/12_k_fold.ipynb (Accessed: December 14, 2022).
13. *Using imbalanced-learn to handle imbalanced text data in NLP* (no date) *Section*. Available at: https://www.section.io/engineering-education/using-imbalanced-learn-to-handle-imbalanced-text-data/ (Accessed: October 29, 2022).
14. Valeti, D. (2021) *Classification using word2vec*, *Medium*. Medium. Available at: https://medium.com/@dilip.voleti/classification-using-word2vec-b1d79d375381 (Accessed: November 17, 2022).