National College of Ireland

Configuration Manual

MSc Research Project
MSC in AI for Business

Raja Muhammad Naveed
Student ID: x23117311

School of Computing
National College of Ireland

Supervisor: Devanshu Anand

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Raja Muhammad Naveed		
Student ID:	x23117311		
Programme:	MSC in AI for Business	**Year:**	2024
Module:	MSC Research Project		
Lecturer:	Devanshu Anand		
Submission Due Date:	12-08-2024		
Project Title:	Configuration Manual		
Word Count:	853 words		
Page Count:	8 Pages		

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: Raja Muhammad Naveed

Date: 12-08-2024

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

Raja Muhammad Naveed
Student ID: x23117311

1 Introduction

The configuration manual provides a detailed guide on the hardware and software requirements, dataset preparation, and implementation of the research. The document outlines how the experiments were carried out; this includes the libraries and packages required as well as steps in preparing datasets, training models, and conducting evaluation.

2 System Configuration

In this section, we will cover the hardware and software Prerequisites needed to test out and finish up project evaluation.

2.1 Hardware Specification

The entire project was implemented using a local machine with the following hardware specifications. This hardware meets the requirements to train and test the models used herein.

Hardware	Build
Processor:	Intel Core i7
RAM:	16 GB
Storage:	512 GB SSD
Operating System:	Windows 11

Figure 1: Hardware Specification

2.1 Software Specification

Implemented in Google Colab to run and use the algorithms, Windows 11 as used (for running these must work you have all required libraries) The necessary tests were carried out by using the following libraries pandas, NumPy, matplotlib, seaborn scikit-learn,textblob textstat wordcloud, nltk.

The details of the software environment used in this project are as follows:
Operating System: Windows 11
Programming Language: Python 3.8.8
IDE: Google Colab
Key Libraries and Versions:

Libaries	Versions
pandas	(version 1.2.4)
numpy	(version 1.20.1)
matplotlib	(version 3.4.1)
seaborn	(version 0.11.1)
scikit-learn	(version 0.24.1)
textblob	(version 0.15.3)
textstat	(version 0.7.1)
wordcloud	(version 1.8.1)
nltk	(version 3.5)

Figure 2: Libraries and Versions

3 Data Set Preparation

The datasets for this project were taken from Kaggle (One for Real News Articles(DataSet_Misinfo_TRUE. csv) and a dataset with fake news articles (DataSet_Misinfo_FAKE. csv). We loaded this data into a Google Colab and connected all datasets (if needed) in one DataFrame, preparing it for the analysis.

Output: A dataset with columns such as text, label and um_text_length This binary label feature indicates 1 for real news and 0 for fake new.

4 Project Implementation

4.1 Dataset Preprocessing

Once we loaded our dataset, some preprocessing was done to make sure the data is clean and ready for training a model. The steps included dealing with missing values and replacing them by empty strings, making new features like text_length, sentiment score using TextBlob adding readability scores taking help form textstat. Outliers removal based on text length to remove very high texts and this would bias the model

```
# Merge the datasets
ALL_NEWS_DF = pd.concat([REAL_NEWS_DF, FAKE_NEWS_DF], ignore_index=True)

# Handle missing values by filling with empty strings
ALL_NEWS_DF['text'] = ALL_NEWS_DF['text'].fillna('').astype(str)

# Add a column for text length
ALL_NEWS_DF['text_length'] = ALL_NEWS_DF['text'].apply(len)

# Display a sample of the dataframe to ensure everything is correct
print(ALL_NEWS_DF.head())
```

```
# Sentiment Analysis
ALL_NEWS_DF['sentiment'] = ALL_NEWS_DF['text'].apply(lambda x: TextBlob(x).sentiment.polarity)

# Readability Scores
ALL_NEWS_DF['readability'] = ALL_NEWS_DF['text'].apply(lambda x: textstat.flesch_reading_ease(x))
```

```
# Outlier Detection and Removal
Q1 = ALL_NEWS_DF['text_length'].quantile(0.25)
Q3 = ALL_NEWS_DF['text_length'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
ALL_NEWS_DF = ALL_NEWS_DF[(ALL_NEWS_DF['text_length'] >= lower_bound) & (ALL_NEWS_DF['text_length'] <= upper_bound)]
```

Figure 3: Data Preproecessing

4.2 Feature Extraction

In order to prepare the text, for machine learning models the Term Frequency Inverse Document Frequency (TF IDF) vectorizer was used. This method converted the text into features by assessing the significance of words, across the dataset.

```
# Initialize the TF-IDF Vectorizer
TFIDF_VECTORIZER = TfidfVectorizer(max_features=5000)

# Fit and transform the text data into numerical features
X_FEATURES = TFIDF_VECTORIZER.fit_transform(ALL_NEWS_DF['text']).toarray()

# Convert labels to numerical values: 1 for 'TRUE', 0 for 'FAKE'
Y_LABELS = ALL_NEWS_DF['label'].apply(lambda x: 1 if x == 'TRUE' else 0).values
```

Figure 4: Feature Extraction using TF-IDF

4.3 Model Training

Three machine learning models were trained on the pre-processed dataset. The Random Forest classifier which consists of multiple decision trees to improve the classification performance, Naïve Bayes classifier which is well suited for text-based classification tasks and a Decision Tree classifier which splits data based on feature values to classify the news articles.

```
# Split the data into training and testing sets
X_TRAIN, X_TEST, Y_TRAIN, Y_TEST = train_test_split(X_FEATURES, Y_LABELS, test_size=0.2, random_state=42)
```

```
# Initialize models
RF_MODEL = RandomForestClassifier(n_estimators=100, random_state=42)
NB_MODEL = MultinomialNB()
DT_MODEL = DecisionTreeClassifier(random_state=42)

# Train models
RF_MODEL.fit(X_TRAIN, Y_TRAIN)
NB_MODEL.fit(X_TRAIN, Y_TRAIN)
DT_MODEL.fit(X_TRAIN, Y_TRAIN)
```

Figure 5: Model Training

5 Evaluation

Three different machine learning models underwent training using the dataset. The Random Forest classifier, employs decision trees to enhance classification accuracy the Naïve Bayes classifier, known for its effectiveness, in text-based classification tasks, and a Decision Tree classifier that divides data according to feature values to categorize news articles.

5.1 Random Forest

Random Forest is a type of machine learning model that combines decision trees to make predictions. During training it creates trees and outputs the common class among these trees. By combining the results of trees Random Forest helps prevent overfitting and improves performance. Here is an example of implementing the Random Forest classifier from the package (Katsaros et al., 2019).

```
# Evaluate models: Random Forest
Y_PRED_RF = RF_MODEL.predict(X_TEST)
rf_probs = RF_MODEL.predict_proba(X_TEST)[:, 1]
rf_auc = roc_auc_score(Y_TEST, rf_probs)
print(f'Random Forest - Accuracy: {RF_MODEL.score(X_TEST, Y_TEST):.2f}, AUC: {rf_auc:.2f}')
print(classification_report(Y_TEST, Y_PRED_RF))
ConfusionMatrixDisplay.from_estimator(RF_MODEL, X_TEST, Y_TEST)
plt.title('Random Forest - Confusion Matrix')
plt.show()
```

Figure 6: Random Forest Implementation Code

5.2 Naïve Bayes

Naive Bayes is a type of Machine Learning model that relies on Bayes theorem. It operates under the assumption that the presence of a feature, in a class is not influenced by the presence of features. This approach has shown effectiveness, in tasks involving text classification. Here we showcase the application of Naïve Bayes through code implementation using MultinomialNB from the sklearn.naive_bayes package (Jain et al., 2022).

```
# Evaluate models: Naive Bayes
Y_PRED_NB = NB_MODEL.predict(X_TEST)
nb_probs = NB_MODEL.predict_proba(X_TEST)[:, 1]
nb_auc = roc_auc_score(Y_TEST, nb_probs)
print(f'Naive Bayes - Accuracy: {NB_MODEL.score(X_TEST, Y_TEST):.2f}, AUC: {nb_auc:.2f}')
print(classification_report(Y_TEST, Y_PRED_NB))
ConfusionMatrixDisplay.from_estimator(NB_MODEL, X_TEST, Y_TEST)
plt.title('Naive Bayes - Confusion Matrix')
plt.show()
```

Figure 7: Naïve Bayes Implementation Code

5.3 Decision Tree

Decision Tree is a machine learning model that uses a flowchart-like structure of decisions and their possible consequences. Decision trees are simple and easy to understand in terms of output, which is why they are often used on problems where the model must be as such. Code to implement the Decision Tree model is presented using sklearn.tree package. (Aphiwongsophon and Chongstitvatana, 2018)

```
# Evaluate models: Decision Tree
Y_PRED_DT = DT_MODEL.predict(X_TEST)
dt_probs = DT_MODEL.predict_proba(X_TEST)[:, 1]
dt_auc = roc_auc_score(Y_TEST, dt_probs)
print(f'Decision Tree - Accuracy: {DT_MODEL.score(X_TEST, Y_TEST):.2f}, AUC: {dt_auc:.2f}')
print(classification_report(Y_TEST, Y_PRED_DT))
ConfusionMatrixDisplay.from_estimator(DT_MODEL, X_TEST, Y_TEST)
plt.title('Decision Tree - Confusion Matrix')
plt.show()
```

Figure 8: Decision Tree Implementation Code

5.4 ROC Curve and AUC Score

Next, the ROC curves were plotted for each model to determine the trade-off between sensitivity and specificity, and from there the AUC score. As it can be seen from the ROC curves in Figure 3, the Random Forest model had the most substantial area under the curve (AUC) showing its merit in discriminating between real and fake news. The AUC for Naive Bayes and Decision Tree were smaller. The R-code for implementing this plot for each model is provided below..

```
# Plot ROC Curves for All Models
rf_fpr, rf_tpr, _ = roc_curve(Y_TEST, rf_probs)
nb_fpr, nb_tpr, _ = roc_curve(Y_TEST, nb_probs)
dt_fpr, dt_tpr, _ = roc_curve(Y_TEST, dt_probs)

plt.figure(figsize=(10, 8))
plt.plot(rf_fpr, rf_tpr, marker='.', label=f'Random Forest (AUC = {rf_auc:.2f})')
plt.plot(nb_fpr, nb_tpr, marker='.', label=f'Naive Bayes (AUC = {nb_auc:.2f})')
plt.plot(dt_fpr, dt_tpr, marker='.', label=f'Decision Tree (AUC = {dt_auc:.2f})')

# Plot the 45 degree line
plt.plot([0, 1], [0, 1], linestyle='--', color='grey')

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend()
plt.grid(True)
plt.show()
```

Figure 9: ROC Curve and AUC Score

6 References

Katsaros, D., Stavropoulos, G. and Papakostas, D. (2019). Which machine learning paradigm for fake news detection? IEEE/WIC/ACM International Conference on Web Intelligence. doi:https://doi.org/10.1145/3350546.3352552.

Aphiwongsophon, S. and Chongstitvatana, P. (2018). *Detecting Fake News with Machine Learning Method*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ECTICon.2018.8620051.

Jain, P., Sharma, S., Monica and Aggarwal, P.K. (2022). Classifying Fake News Detection Using SVM, Naive Bayes and LSTM. [online] IEEE Xplore. doi:https://doi.org/10.1109/Confluence52989.2022.9734129.