National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Shiva Vasineni
Student ID: 23201274

School of Computing
National College of Ireland

Supervisor: Dr. William Clifford

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. …… Shiva Vasineni ……

Student ID: ……23201274……..……

Programme: …… MSc in Data Analytics …… **Year:** ……2024-2025…..

Module: ……MSc Research Project……..……

Lecturer: ……Dr. William Clifford……..……

Submission Due Date: ……29 January 2025……..……

Project Title: …… Deep Learning Approaches Identifying Fake Reviews in E commerce Platforms ……..……

Word Count: ……385…… **Page Count:** ……6……..……

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: ……Shiva Vasineni……

Date: ……29 January 2025……

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

Shiva Vasineni
Student ID: 23201274

1 Overview

The code is aimed at detecting fake reviews using various machine learning like decision tree, random forest etc and deep learning models like RNN, LSTM, CNN+BiLSTM models. It involves preprocessing textual data, exploring the dataset, training models, and evaluating their performance.

2 Steps Involved in code

1. **Data Loading and Cleaning**:
    - Load datasets containing reviews and labels (original vs. fake).
    - Clean and preprocess text data by removing special characters, converting to lowercase, and removing stopwords.
2. **Data Exploration**:
    - Analyzing the distribution of review lengths and labels.
    - Visualizing the important insights using plots (e.g., word clouds, review length distributions).
3. **Text Vectorization**:
    - Convert text data into numerical format using **TF-IDF Vectorizer**.
4. **Model Training and Tuning**:
    - Use classical machine learning models like:
        - **Decision Tree Classifier**
        - **Random Forest Classifier**
        - **Naive Bayes**
    - Tune hyperparameters using GridSearchCV.
    - Train deep learning models including:
        - **Simple RNN**
        - **LSTM**
        - **CNN-BiLSTM**
5. **Evaluation**:
    - Evaluate models using metrics like accuracy, precision, recall, and F1-score on the validation and test datasets.

6. System requirements:

- Multi-core CPU processor, Having RAM of at least 8GB, with using visual studio or Jupyter notebook or google colab.
- Sufficient space for storing datasets, intermediate files, and libraries (~5 GB for Python environment and dependencies).
- NVIDIA GPU with CUDA support for accelerating deep learning models.

3 Libraries Required

All the following installations are required for running code

1. pip install pandas
2. pip install matplotlib
3. pip install seaborn
4. pip install nltk
5. pip install scikit-learn
6. pip install scipy
7. pip install numpy
8. pip install torch

Your second section. Change the header and label to something appropriate.

4 Code snippets

1. Add path for test and train files in code

```
# Load the train and test datasets
train_file_path = './train_data.csv'
test_file_path = './test_data.csv'
```

2. Checking for outliers in data

```
# Define thresholds for review length
min_review_length = 20
max_review_length = 1000

# Filter out outlier reviews from the train dataset
filtered_train_data_no_outliers = filtered_train_data[
    (filtered_train_data['review_length'] >= min_review_length) &
    (filtered_train_data['review_length'] <= max_review_length)
]

# Filter out outlier reviews from the test dataset
filtered_test_data_no_outliers = filtered_test_data[
    (filtered_test_data['review_length'] >= min_review_length) &
    (filtered_test_data['review_length'] <= max_review_length)
]
```

3. Converting text to embeddings

```
# Preprocessing strategy 1: Vectorizing cleaned text using TF-IDF
tfidf = TfidfVectorizer(max_features=1000, stop_words='english')
```

4. Dividing training data to validation and train split

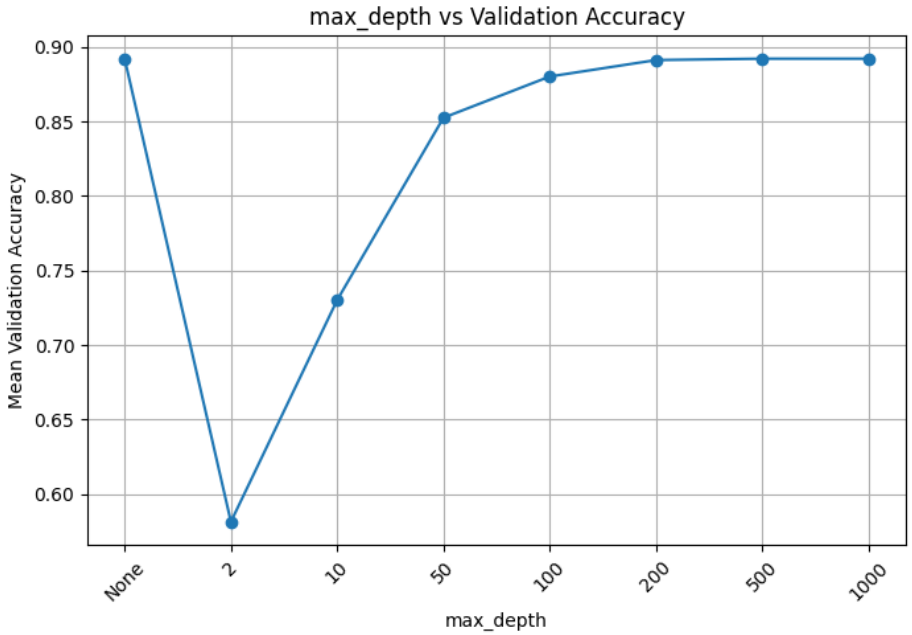
```
# Split into training and validation sets (e.g., 80% train, 20% validation)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=SEED)
```

5. Training Decision Tree model with Hyperparamters

```
# Parameter grid with an additional hyperparameter: criterion
param_grid = {
    'max_depth': [None, 2, 10, 50, 100, 200, 500, 1000],
    'min_samples_split': [2, 5, 10, 20, 30, 50],
    'criterion': ['gini', 'entropy']
}

# Train GridSearchCV on training data
dt = DecisionTreeClassifier(random_state=42)
grid_search = GridSearchCV(estimator=dt, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
```

6. Maximum depth vs validation accuracy for decision tree model



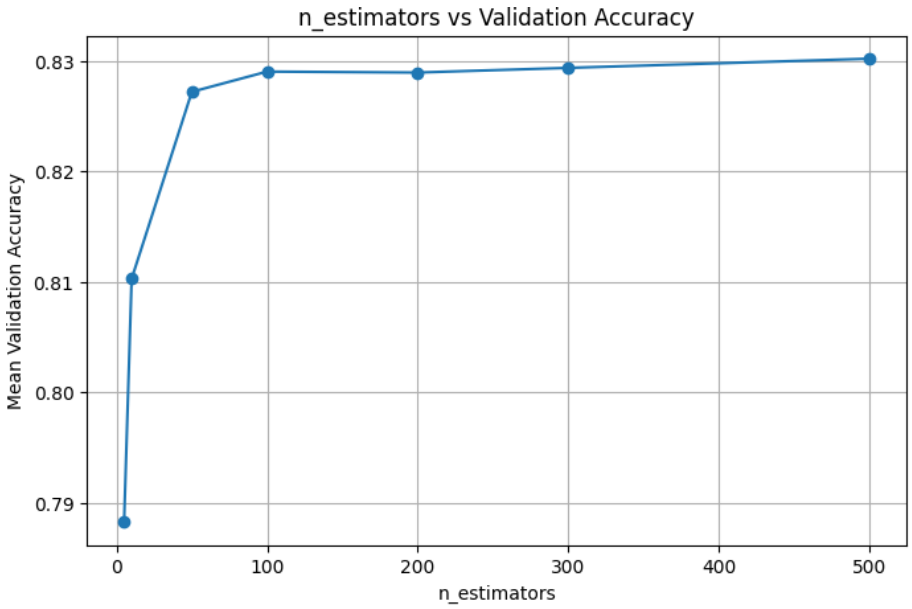
7. Training ramdom forest model with hyper paramters

```
# Define the parameter grid
param_grid = {
    'n_estimators': [5, 10, 50, 100, 200, 300, 500],
    'max_depth': [2, 3, 5, 10, 20, 30, 40, 50, 60],
}

# Initialize the Random Forest model
rf = RandomForestClassifier(random_state=42)

# Train GridSearchCV on training data
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
```

8. Hyperparamter plot of random forest



9. Defining RNN model

```
# Define the Simple RNN model
class RNNModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, output_size=1):
        super(RNNModel, self).__init__()
        self.rnn = nn.RNN(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x, _ = self.rnn(x)  # RNN output: (batch_size, sequence_length, hidden_size)
        x = x[:, -1, :]  # Take the last hidden state
        x = self.fc(x)
        x = self.sigmoid(x)
        return x
```

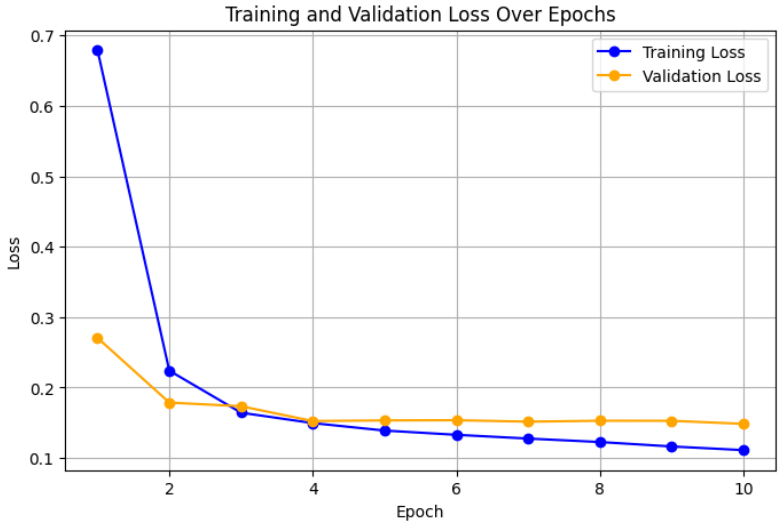
10. Training and validation loss of RNN

11. Intialising CNN+BiLSTM model

```python
#define the CNN + BiLSTM model
class CNNBiLSTMModel(nn.Module):
    def __init__(self, input_size):
        super(CNNBiLSTMModel, self).__init__()
        self.conv1 = nn.Conv1d(in_channels=1, out_channels=128, kernel_size=3, padding=1)
        self.pool = nn.MaxPool1d(kernel_size=2)
        self.bilstm = nn.LSTM(input_size=128, hidden_size=64, num_layers=1, batch_first=True, bidirectional=True)
        self.fc1 = nn.Linear(64 * 2 * (input_size // 2), 64)  # Adjusted based on pooling
        self.fc2 = nn.Linear(64, 1)
        self.dropout = nn.Dropout(0.5)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = x.unsqueeze(1)
        x = self.conv1(x)
        x = self.pool(x)
        x = x.permute(0, 2, 1)
        x, _ = self.bilstm(x)
        x = x.contiguous().view(x.size(0), -1)
        x = self.fc1(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = self.sigmoid(x)
        return x
```

12. Training and validation Loss of CNN+BiLSTM model



References

Géron, A., 2022. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.".

https://pandas.pydata.org/docs/

https://docs.jupyter.org/en/latest/

https://pytorch.org/docs/stable/index.html