National College of Ireland

Deep Learning-Based Approaches for Detecting AI-Generated Fake Reviews in Online Platforms

MSc Research Project
Data Analytics

Ashutosh Bangari
Student ID: 23412992

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ashutosh Bangari
Student ID:	23412992
Programme:	Data Analytics **Year:** 2025
Module:	Research Practicum
Lecturer:	Vladimir Milosavljevic
Submission Due Date:	11 Dec 2025
Project Title:	Deep Learning-Based Approaches for Detecting AI-Generated Fake Reviews in Online Platforms
Word Count:	1237 **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: Ashutosh Bangari

Date: 10/12/2025

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):	

Deep Learning-Based Approaches for Detecting AI-Generated Fake Reviews in Online Platforms

Ashutosh Bangari
23412992

1. Introduction

This configuration manual outlines the hardware and software environment required to reproduce the experimental workflow conducted in this thesis. The study involves a combination of traditional machine learning techniques and transformer-based natural language processing models; therefore, the system specifications prioritise efficient preprocessing, vectorisation, and model training. The hardware configuration focuses on ensuring adequate computational capacity for TF-IDF matrix generation, classical ML execution, and transformer-based embedding extraction using DistilBERT. The software environment specifies the operating system, interpreter version, and supporting libraries required to ensure compatibility, reproducibility, and consistent runtime performance across all stages of the pipeline.

2. System Specifications

Hardware requirements

- **CPU:** Modern multi-core CPU (recommended: 4+ cores) for preprocessing, TF-IDF generation, and classical ML training.
- **GPU:** NVIDIA CUDA-capable GPU recommended for DistilBERT embedding extraction; CPU-only execution is possible but slower for transformer embedding batches.
- **RAM:** Minimum 8 GB; recommended 16 GB+ to comfortably handle TF-IDF matrices and intermediate arrays (e.g., DistilBERT embedding matrices of shape (32329, 768) and (8083, 768)).
- **Storage:** At least 2-5 GB free space, considering dataset files, cached transformer models, and saved .npy embedding arrays.

Software requirements

- **Operating System:** Windows 10/11, Linux (Ubuntu 20.04+), or macOS; Linux is typically preferred for GPU/CUDA workflows.
- **Python version:** Python 3.9+ recommended for compatibility with Transformers, PyTorch, and TensorFlow.
- **Environment:** Conda or venv recommended to isolate dependencies; Jupyter/Colab environment supported (the provided workflow matches a Colab-style execution).

- **CUDA/cuDNN (if GPU used):** CUDA toolkit and cuDNN versions must match the installed PyTorch build; TensorFlow GPU compatibility must be ensured if training deep networks on GPU.

3. Required Libraries

The project uses a combination of data science tooling, NLP utilities, classical ML frameworks, deep learning libraries, and transformer tooling. NumPy and Pandas support numerical computation and tabular manipulation. Matplotlib and Seaborn are used for EDA plots and evaluation visualizations. NLTK provides tokenization, stop word removal, and lemmatization for text normalization. Scikit-learn is used for train-test splitting, label encoding, TF-IDF vectorization, GridSearchCV hyperparameter tuning, and evaluation metrics. TensorFlow/Keras provides GRU-based neural models. PyTorch and Hugging Face Transformers power DistilBERT tokenization and contextual embedding extraction, enabling a hybrid design where the transformer produces 768-dimensional representations that feed a GRU classifier.

Installation commands (typical environment setup) are as follows:

A. **pip install numpy pandas matplotlib seaborn scikit-learn nltk wordcloud tqdm**
B. **pip install torch transformers**
C. **pip install tensorflow**
D. **pip install xgboost**

Utility modules include re and string for text cleaning operations, and warnings to suppress non-critical runtime warnings during experimentation.

4. Dataset description

The data applied in this research is comprised of customer reviews of the different category and each record has four primary measures, namely, category, rating, label, and text. The column of categories determines the product domain, with the rating column giving feedback in numerical form between 1 and 5 which depict the level of satisfaction among the users. The label field tells whether the review is a CG (computer-generated) or an OR (human-written) text so that a binary classification can be used to detect the text created by AI. The actual review text, which serves as the main input in model training and analysis is contained in the text column. This data has a fair sample of real and AI-written reviews, with the variety of linguistic styles, tones, and the intensity of sentiments. It is specifically appropriate to test the hybrid TransformerGRU model because it can extract semantic features of the complex sentences and learn temporal dependency in sequential text data, which can be used to effectively test the ability of the model in authenticity in real-life review and sentiment analysis tasks.

5. Models Implemented

The modelling strategy progresses in stages to enable fair comparison and clearly justify the final approach. First, classical ML models are trained on TF-IDF features: a Decision Tree provides a baseline, Random Forest improves stability through assembling, and XGBoost boosts performance via gradient-based tree optimization. Next, a GRU-based deep learning model is trained on padded token sequences to capture word-order and sequential patterns missed by bag-of-words methods. Finally, a hybrid DistilBERT + GRU architecture uses DistilBERT as a frozen feature extractor to generate 768-dimensional contextual embeddings, which a GRU classifier then learns from, achieving strong semantic modelling without the cost of full transformer fine-tuning.

6. Code snippets and plots:

1) Dataset information

```
# Check basic info
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40432 entries, 0 to 40431
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   category  40432 non-null  object
 1   rating    40432 non-null  int64 
 2   label     40432 non-null  object
 3   text_     40432 non-null  object
dtypes: int64(1), object(3)
memory usage: 1.2+ MB
None
```

2) Splitting data

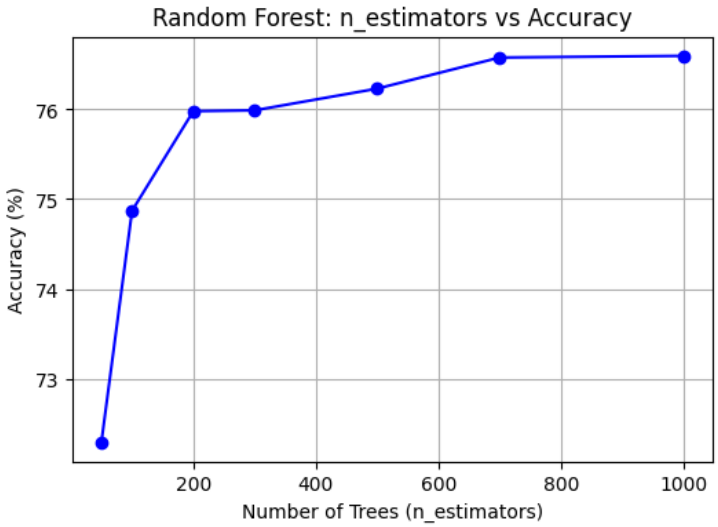
```
from sklearn.model_selection import train_test_split

X=df['clean_text']
y=df['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

3) Encoding

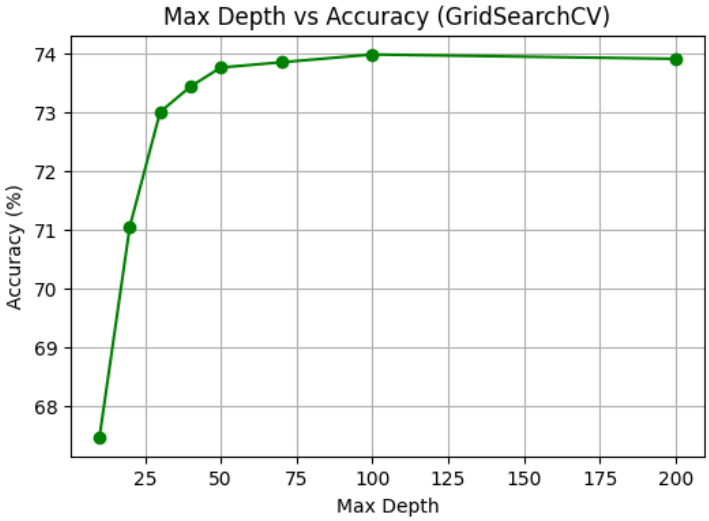
```python
#  Encode labels
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train_enc = le.fit_transform(y_train)
y_test_enc = le.transform(y_test)
num_classes = len(le.classes_)
print("Number of classes:", num_classes)
```

```
Number of classes: 2
```

4) Hyper parameter plot for random forest



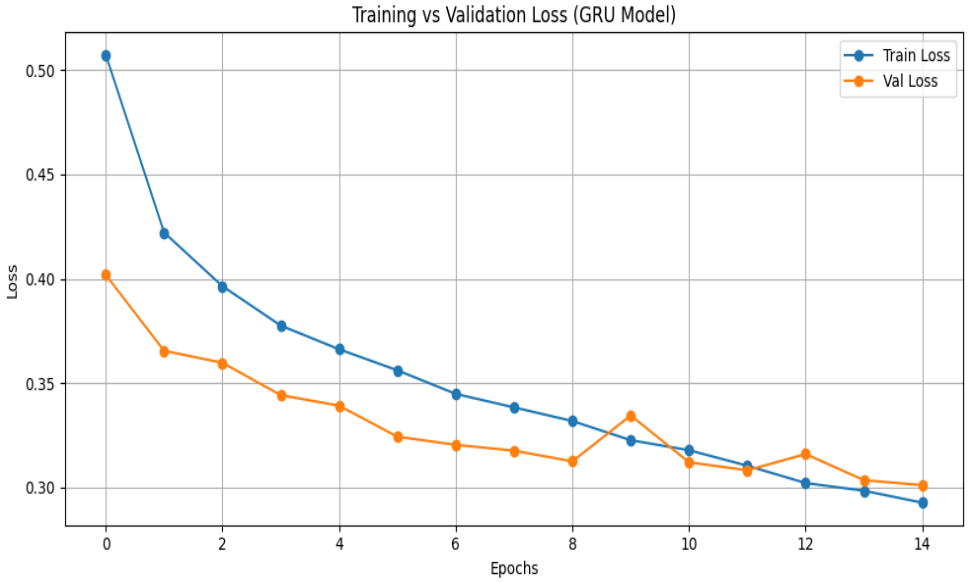
5) Hyper parameter plot for decision tree



6) Distilbert + GRU

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense, Dropout, BatchNormalization, Input, Reshape
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score

# Build GRU Model
model = Sequential([
    Input(shape=(768,)),
    Reshape((1, 768)),
    GRU(128, return_sequences=False),
    Dropout(0.4),
    BatchNormalization(),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')])

# Compile model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=3e-4),
    loss='binary_crossentropy',
    metrics=['accuracy'])
```

7) Loss plot for DistilBERT + GRU



8) Comparison of evaluation metrics

Table 1. Model Performance Comparison

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	74.18%	74.19%	74.18%	74.18%
Random Forest	77.32%	77.42%	77.32%	77.30%
XGBoost	82.15%	82.21%	82.15%	82.14%
GRU	83.36%	84.93%	83.36%	83.17%
DistilBERT + GRU	86.12%	86.13%	86.12%	86.12%

7. Conclusion:

This project is replicable and methodologically structured: data ingestion, validation, deduplication, exploratory analysis, text cleaning, feature engineering, model training, and standardized evaluation are implemented as a coherent workflow. The hybrid DistilBERT + GRU model is the best-performing approach, achieving 86.12% accuracy, because it combines contextual embeddings (capturing semantic nuance and syntactic structure) with GRU-based sequential modelling (capturing patterns that separate AI-generated from human-written text). Importantly, using DistilBERT as a frozen feature extractor improves computational efficiency relative to full transformer fine-tuning while retaining strong representational power.
The pipeline can be extended in future. It is able to add domain adaptation (category-specific training), Multi projecting multilingual, more robust transformer backbones (with attentive efficiency regulators) and clarification measures to enhance trust in forecasts. It can also be applied in the business world in integrations where authenticity review should be reliable in changing AI-generated textual quality.

References:

1. https://seaborn.pydata.org/
2. https://huggingface.co/docs/transformers/en/index
3. https://www.nltk.org/
4. https://www.tensorflow.org/
5. https://scikit-learn.org/stable/
6. https://numpy.org/
7. https://pandas.pydata.org/
8. https://huggingface.co/docs/transformers/model_doc/distilbert