

## ▼ Project 2:

Practical\_Image\_Processing\_and\_Natural\_Language\_Processing\_Second Trimester 2026

Dr.Hajer Saleh

Faisal Ahmed Almokayed - 253001589

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

# Use the kagglehub client library to attach Kaggle resources like competitions, datasets, and models to your session
# Learn more about kagglehub: https://github.com/Kaggle/kagglehub/blob/main/README.md

import kagglehub
# kagglehub.dataset_download('<owner>/<dataset-slug>')

/kaggle/input/datasets/shanza30/humanvsai-2026-synthetic-text-detection-dataset/HumanVsAI-2026.csv
```

## ▼ import Libraries and Configuration

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
import re
import nltk
import spacy
import warnings, re, string, time
warnings.filterwarnings('ignore')
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk import ngrams
from wordcloud import WordCloud
from collections import Counter
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

```

from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

```

```

# Download NLTK resources (only first time)
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

```

```

[nltk_data] Downloading package punkt to /usr/share/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /usr/share/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
True

```

```

import tensorflow as tf
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import (Embedding, LSTM, GRU, Dense, Dropout,
                                     Bidirectional, GlobalMaxPooling1D,
                                     Conv1D, MaxPooling1D, Flatten, Input)
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam

```

## Loading Dataset

```

df = pd.read_csv('/kaggle/input/datasets/shanza30/humanvsai-2026-synthetic-text-detection-dataset/HumanVsAI-2026.csv')
print('=' * 55)
print('      DATASET OVERVIEW — HumanVsAI-2026')
print('=' * 55)
print(f' Total Samples      : {len(df):,}')
print(f' Total Features      : {df.shape[1]:,}')
print(f' Human Samples       : {(df.label=="Human").sum():,}')
print(f' AI Samples          : {(df.label=="AI").sum():,}')
print(f' Missing Values      : {df.isnull().sum().sum():,}')
print(f' Duplicate Rows      : {df.duplicated().sum():,}')
print('=' * 55)
df.head()

```

```

=====
      DATASET OVERVIEW — HumanVsAI-2026
=====
Total Samples      : 5,000
Total Features     : 7
Human Samples      : 2,462
AI Samples         : 2,538
Missing Values     : 0
Duplicate Rows     : 0
=====

```

|   | id | text                                               | label | source   | complexity_score | grammar_errors_count | sentence_length |
|---|----|----------------------------------------------------|-------|----------|------------------|----------------------|-----------------|
| 0 | 1  | i think technology is kinda har but i'll try anywa | Human | research | 0.49             | 2                    | 10              |
| 1 | 2  | kal ma machine learning krna bhool gaya lol        | Human | essay    | 0.43             | 3                    | 8               |
| 2 | 3  | Deep learning techniques have improved technol...  | AI    | blog     | 0.64             | 1                    | 7               |
| 3 | 4  | Machine learning models are widely used in stu...  | AI    | blog     | 0.71             | 1                    | 11              |
| 4 | 5  | Artificial intelligence is transforming machin...  | AI    | tweet    | 0.85             | 0                    | 10              |

## Evaluation Setup

```

def evaluate_model(model_name, y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')
    report = classification_report(y_true, y_pred)

    print("="*60)
    print("TEST SET EVALUATION RESULTS")

```

```

print("="*60)
print(f"Model Name      : {model_name}")
print(f"Overall Accuracy : {accuracy:.4f}  ({accuracy*100:.2f}%)")
print(f"Precision        : {precision:.4f}")
print(f"Recall            : {recall:.4f}")
print(f"F1 Score         : {f1:.4f}")
print("\nClassification Report:\n")
print(report)
print("="*60)

```

## Setup Confusion Matrix

```

def plot_confusion_matrix(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6,4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Confusion Matrix - {model_name}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

```

## Preprocessing

lowercase, remove punctuation, remove numbers, remove stopwords, and lemmatization

```

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

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    if not isinstance(text, str):
        return ""

    # 1. Lowercase
    text = text.lower()

    # 2. Remove URLs
    text = re.sub(r'http\S+|www.\S+', '', text)

    # 3. Remove HTML tags
    text = re.sub(r'<.*?>', '', text)

    # 4. Remove special characters & numbers
    text = re.sub(r'^a-zA-Z\s]', '', text)

    # 5. Tokenize
    tokens = word_tokenize(text)

    # 6. Remove stopwords
    tokens = [t for t in tokens if t not in stop_words]

    # 7. Lemmatize
    tokens = [lemmatizer.lemmatize(t) for t in tokens]

    # 8. Reconstruct
    clean_text = " ".join(tokens)

    return clean_text

```

```

df = df.dropna(subset=["text"])
df["clean_text"] = df["text"].apply(preprocess_text)
df["clean_text"].head()

```

```

0      think technology kinda har il try anywa
1      kal machine learning krna bhool gaya lol
2      deep learning technique improved technology si...
3      machine learning model widely used studying be...
4      artificial intelligence transforming machine l...
Name: clean_text, dtype: object

```

## ▼ Data Splitting (70 / 30)

70 % for training and 30 % for Testing

```
from sklearn.model_selection import train_test_split

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

# 70% train, 30% test
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.30,
    random_state=42,
    stratify=y
)

print("Train size:", len(X_train))
print("Test size :", len(X_test))
```

Train size: 3500  
Test size : 1500

## ▼ Before VS After Preprocessing

```
print("\n 📁 Before vs After Preprocessing:")
print("-" * 110)
for i in range(3):
    original = df["text"].iloc[i]
    cleaned = df["clean_text"].iloc[i]
    print(f"{'ORIGINAL':<10}: {original:<50} | {'CLEANED':<10}: {cleaned}")
    print("-" * 120)
```

📁 Before vs After Preprocessing:

|          |                                                                    |  |         |                                            |
|----------|--------------------------------------------------------------------|--|---------|--------------------------------------------|
| ORIGINAL | : i think technology is kinda har but i'l try anywa                |  | CLEANED | : think technology kinda har il try anywa  |
| ORIGINAL | : kal ma machine learning krna bhool gaya lol                      |  | CLEANED | : kal machine learning krna bhool gaya lol |
| ORIGINAL | : Deep learning techniques have improved technology significantly. |  | CLEANED | : deep learning technique improved         |

## ▼ Vectorization

```
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

## ▼ FEATURE REPRESENTATION METHODS (ML)

Logistic Regression, Naive Bayes, Random Forest, Support Vector Machine, Gradient Boosting

```
models = {
    "Logistic Regression": LogisticRegression(max_iter=200),
    "Naive Bayes": MultinomialNB(),
    "Random Forest": RandomForestClassifier(n_estimators=200, random_state=42),
    "Support Vector Machine": SVC(kernel='linear', probability=True),
    "Gradient Boosting": GradientBoostingClassifier()
}
```

## ▼ Train Models

```
trained_models = {}

for name, model in models.items():
```

```
print(f"\n🟡 Training {name}...")
model.fit(X_train_vec, y_train)
trained_models[name] = model
```

- 🟡 Training Logistic Regression...
- 🟡 Training Naive Bayes...
- 🟡 Training Random Forest...
- 🟡 Training Support Vector Machine...
- 🟡 Training Gradient Boosting...

## ✖ Evaluation

```
def collect_metrics(model_name, y_true, y_pred):
    return {
        "Model": model_name,
        "Accuracy": accuracy_score(y_true, y_pred),
        "Precision": precision_score(y_true, y_pred, average='weighted'),
        "Recall": recall_score(y_true, y_pred, average='weighted'),
        "F1 Score": f1_score(y_true, y_pred, average='weighted')
    }
```

```
results = []

for name, model in trained_models.items():
    print(f"\n🟡 Evaluating {name}...")

    y_pred = model.predict(X_test_vec)

    # Your existing evaluation function
    evaluate_model(name, y_test, y_pred)

    # Confusion matrix
    plot_confusion_matrix(y_test, y_pred, name)

    # Add metrics to results list
    results.append(collect_metrics(name, y_test, y_pred))
```

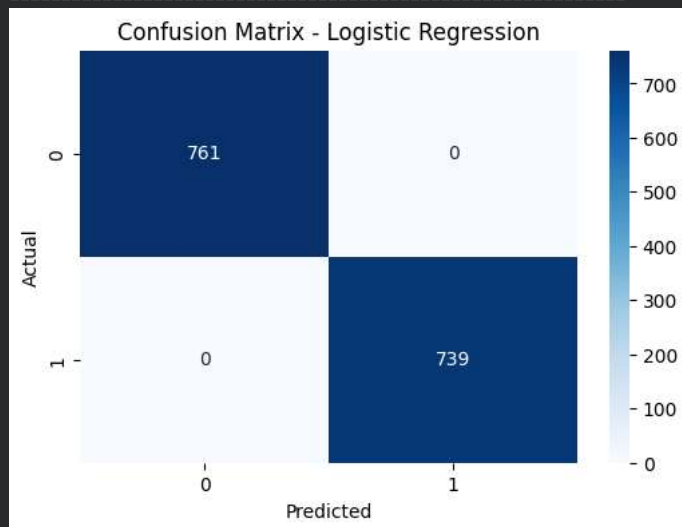
Evaluating Logistic Regression...

#### TEST SET EVALUATION RESULTS

Model Name : Logistic Regression  
Overall Accuracy : 1.0000 (100.00%)  
Precision : 1.0000  
Recall : 1.0000  
F1 Score : 1.0000

Classification Report:

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| AI           | 1.00      | 1.00   | 1.00     | 761     |
| Human        | 1.00      | 1.00   | 1.00     | 739     |
| accuracy     |           |        | 1.00     | 1500    |
| macro avg    | 1.00      | 1.00   | 1.00     | 1500    |
| weighted avg | 1.00      | 1.00   | 1.00     | 1500    |



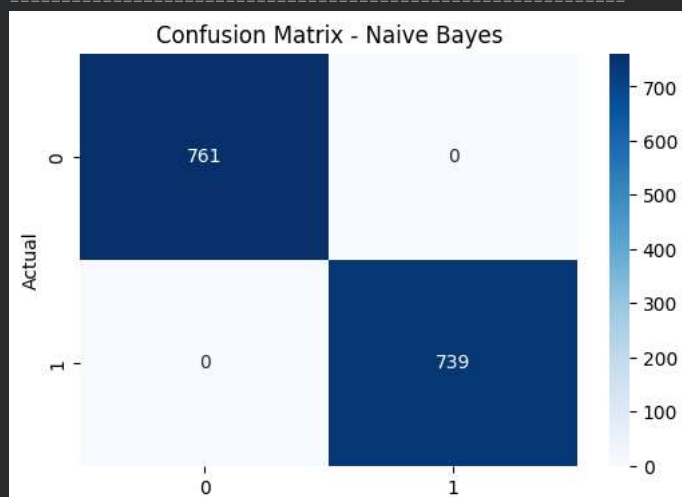
Evaluating Naive Bayes...

#### TEST SET EVALUATION RESULTS

Model Name : Naive Bayes  
Overall Accuracy : 1.0000 (100.00%)  
Precision : 1.0000  
Recall : 1.0000  
F1 Score : 1.0000

Classification Report:

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| AI           | 1.00      | 1.00   | 1.00     | 761     |
| Human        | 1.00      | 1.00   | 1.00     | 739     |
| accuracy     |           |        | 1.00     | 1500    |
| macro avg    | 1.00      | 1.00   | 1.00     | 1500    |
| weighted avg | 1.00      | 1.00   | 1.00     | 1500    |



## Predicted

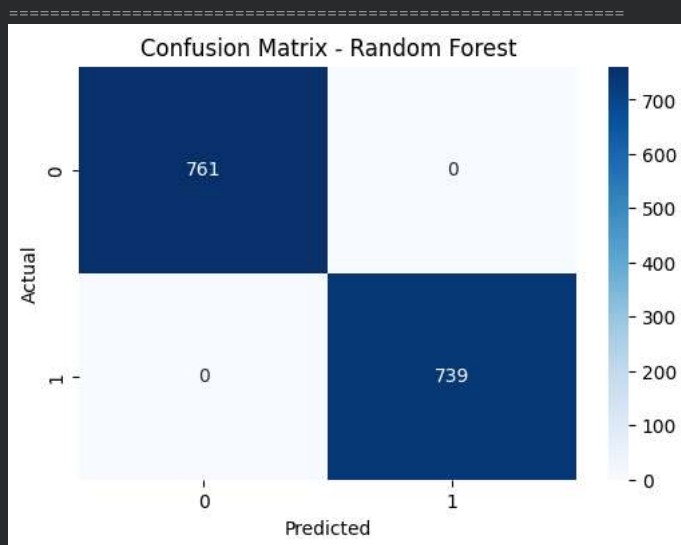
🔍 Evaluating Random Forest...

### TEST SET EVALUATION RESULTS

```
Model Name      : Random Forest
Overall Accuracy : 1.0000 (100.00%)
Precision        : 1.0000
Recall           : 1.0000
F1 Score         : 1.0000
```

Classification Report:

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| AI           | 1.00      | 1.00   | 1.00     | 761     |
| Human        | 1.00      | 1.00   | 1.00     | 739     |
| accuracy     |           |        | 1.00     | 1500    |
| macro avg    | 1.00      | 1.00   | 1.00     | 1500    |
| weighted avg | 1.00      | 1.00   | 1.00     | 1500    |



🔍 Evaluating Support Vector Machine...

### TEST SET EVALUATION RESULTS

```
Model Name      : Support Vector Machine
Overall Accuracy : 1.0000 (100.00%)
```

```
results_df = pd.DataFrame(results)
results_df = results_df.sort_values(by="Accuracy", ascending=False)
results_df.style.background_gradient(cmap="Blues")
```

| Model                    | precision | Accuracy | Recall   | F1 Score |
|--------------------------|-----------|----------|----------|----------|
| 0 Logistic Regression    | 1.000000  | 1.000000 | 1.000000 | 1.000000 |
| 1 Naive Bayes            | 1.000000  | 1.000000 | 1.000000 | 1.000000 |
| 2 Random Forest          | 1.000000  | 1.000000 | 1.000000 | 1.000000 |
| 3 Support Vector Machine | 1.000000  | 1.000000 | 1.000000 | 1.000000 |
| 4 Gradient Boosting      | 1.000000  | 1.000000 | 1.000000 | 1.000000 |
| weighted avg             | 1.00      | 1.00     | 1.00     | 1500     |

```
if results_df.empty:
    print("⚠️ No results found. Check your evaluation loop.")
else:
    best_model_row = results_df.iloc[0]
    best_model_name = best_model_row["Model"]

    print("=====")
    print("🏆 BEST MODEL SUMMARY")
    print("=====")
```

## Best Model Report