Project Title: Detection of AI-Generated Arabic Text: A Data Mining Approach

Course: MSIS-822 Advanced Data Analytic Techniques

Student Name: Abdulmajeed Alqahtani

Student ID: 4714235

Date: December 14, 2025

Content Table

Abstract.

Large Language Models (LLMs) powered to near-extent by GPT-4, Llama, and Jais have democratized content creation but heightened academic dishonesty and disinformation risks. Detecting machine-generated text is especially difficult given the rich morphology and intricate syntactic structures of the Arabic language. This study presents a data mining pipeline to differentiate human-written and AI-generated Arabic abstracts using a stylometric approach. We created five interpretable features with a large, balanced dataset (e.g. 16,776 samples) available from the KFUPM-JRCAI/arabic-generated-abstracts repository: Word Length Distribution, Characters per Paragraph, Command Frequency, Average Sentence Length and a custom Formality Score.

We introduced and tested three supervised machine learning models: Logistic Regression as a baseline, Support Vector Machine (SVM), and Random Forest. The Random Forest classifier demonstrated the highest accuracy of 99.08% through experiments. Notably, the simple Logistic Regression baseline made an impressive 97.26%, slightly outperforming the SVM (97.23%). These findings indicate that lightweight stylometric attributes are excellent fingerprints for identifying AI-generated content.

1.Introduction.

1.1 Motivation: Text production was transformed through the use of Artificial Intelligence. Although the tools for detecting English have developed, it appears that detection of Arabic has not been covered to the same extent in a survey conducted within the country. Arabic writing systems, while more powerful, tend to rely on high-dimensional structures that capture their information and are rarely well-maintained. Indeed, with its complex diacritics system or the roots derived from it, not to mention the diversity in dialects—all of which make simple keyword matching ineffective—Arabic can neither be described nor understood as "standard". There is a serious need for computational methods which can 'fingerprint' writing styles of AI without using computationally expensive deep learning architectures. AI models are designed to minimize perplexity, yielding grammatically correct text that is style-aligned and unhelpful. But this optimizing logic-method often generates subtle statistical artifacts. Human writing is stochastic — it exhibits creativity for a reason, sentence structure is chaotic and ranges widely in depth vocabulary — that the AI cannot duplicate in a perfect manner.

1.2 Objectives: The task of this project is to explore the potential role of Arabic AI detection through the following three processes:

1.Building to Pipeline: Creating a pipeline from raw data to model deployment in the data mining process from data entry till model deployment.

2.Feature Engineering: Develop five stylometric (or stylistics) features that will represent the structural rigidity of AI-based techniques, based on Arabic linguistics.

3.Interpretability: Considering importance of various features, to see why the model defines a text to be AI, especially from the aspect of sentence structure dynamics.

2.Dataset Description.

2.1 Data source:
The project is based off the KFUPM-JRCAI/arabic-generated-abstracts dataset which is another state of the art pool for Arabic AI detection. It has original human abstracts in pair with AI-created abstracts from different models (OpenAI GPT, Llama-2, Jais and Allam.(

2.2 Data Preparation:
and Balancing Data were pooled in several different generation methods (by_polishing, from_title, etc) to achieve a strong and non biased judgment. One of the major steps in our approach was Class Balancing.

1.Status: The dataset had orders of magnitude more AI produced samples compared to human samples.

2.Balancing Method: We used a random undersampling method on the majority class (AI) to equal the exact number of the minority class (Human.(

3.Final Statistics:
Total Samples: 16,776.
Human (Class 0): 8,388 data (50%).
AI (Class 1): 8,388 data (50%)

The dataset was further divided into an 80% Training Set (13,420 samples) and a 20% Testing Set (3,356 samples).

3. Methodology: Stylometric Feature Engineering

Unlike deep learning approaches that treat text as opaque embeddings, our methodology relies on Explainable Stylometry. We hypothesized that AI models leave statistical footprints in the structure of the text. We engineered five features applied to the Original Raw Text (to preserve punctuation and layout).

3.1 Word Length Frequency Distribution (Feature 15)

Concept: This feature quantifies complexity of vocabulary.
Logic: We compute the Mean and the Standard Deviation of character counts for every word.
Hypothesis: AI models prefer high-probability words, leading to a uniform average length. People have a range of vocabulary depth (such as very short words and very long words), providing a higher variance.

3.2 Average Characters per Paragraph (Feature 38)

Concept: This concept measures information density and layout structure. Logic: Average = Total Characters / Count of Newline Blocks.
Hypothesis: AI-generated abstracts often appear as single, dense blocks or perfectly balanced paragraphs. Human researchers often employ irregular paragraph structures to separate logical sections.

3.3 Number of Commands / Imperative Verbs (Feature 61)

Concept: Analyzing sentence structure as a syntax of "Instructional Tone."
Logic: A dictionary lookup counts Arabic imperative verbs (e.g., "Look", "Note", "Compare").
Hypothesis: AI models trained on instruction-following data may accidentally adopt a "tutorial" style in abstracts. Usually humans adopt the passive or descriptive voice.

3.4 Average Sentence Length (Feature 84)

Concept: Measures the rhythmic structure of the text.
Logic: The text is segmented into sentences using punctuation (periods, question marks). We calculate the mean number of words per sentence. Hypothesis: This has the strongest predictive power. AI models pay attention to "readability" and "coherence," and their sentences are made to be at a uniform or standardized length. Human writing is described as "burstiness," which means it can be short, punchy bursts of sentences, then long and complicated compound sentences.

3.5 Formality Score (Feature 107)

Concept: It is calculated on a scale of "Modern Standard Arabic (MSA)" strictness.
Logic: Score = (Formal Indicators − Informal Indicators) / Total Words. Formal: Words beginning with "Al-", ending with "iyya" or words greater than 6 letters.
Informal: Personal pronouns (e.g., I, We), words shorter than 3 letters. Hypothesis: AI is hyper-correct and formal. Humans could add stylistic nuances or personal pronouns.

3.6 Machine Learning Models:

To evaluate the discriminative power of our features, we implemented three distinct classifiers:

1. Logistic Regression (Baseline): Used to establish a performance benchmark and test linear separability.
2. Support Vector Machine (SVM): A robust linear classifier.
3. Random Forest: An ensemble learning method selected to capture non-linear interactions between features.

4. Implementation Details

This section documents the technical implementation using Python. The code is modularized to ensure reproducibility.

4.1 Library Initialization: We utilize nltk for linguistic processing and sklearn for modeling

```
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.tokenize import word_tokenize, sent_tokenize
from datasets import load_dataset
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix

 #Ensure Arabic tokenizers are available
try:
   nltk.data.find('tokenizers/punkt('
except LookupError:
   nltk.download('punkt')
```

4.2 Data Acquisition Logic: The load_full_data function manages the retrieval and balancing of the 16,000+ samples.

```
def load_full_data:()
   dataset = load_dataset("KFUPM-JRCAI/arabic-generated-abstracts("
   dfs[] =
 #    Aggregate all data splits
   for split in ['by_polishing', 'from_title', 'from_title_and_content:['
      if split in dataset:
         dfs.append(pd.DataFrame(dataset[split(([
   df_merged = pd.concat(dfs, ignore_index=True(

 #    Separation and Balancing Logic
   human = df_merged['original_abstract'].dropna().tolist()
   ai[] =
   for col in df_merged.columns:
      if 'generated_abstract' in col:
         ai.extend(df_merged[col].dropna().tolist(()

   min_len = min(len(human), len(ai((
 #    Return perfectly balanced DataFrame
   return pd.DataFrame({'text': human[:min_len] + ai[:min_len ,[
                      'label': [0]*min_len + [1]*min_len}).sample(frac=1)
```

4.3 Feature Engineering Code:The core features are implemented as functions applied to the dataframe.

```
 #F15: Word length-frequency distribution
def feat_word_len_dist(text:(
    if not isinstance(text, str) or not text.strip(): return 0, 0
    words = word_tokenize(text(
    if not words: return 0, 0
    lengths = [len(w) for w in words[
    return np.mean(lengths), np.std(lengths(

 #F38: Average Characters per Paragraph
def feat_chars_per_paragraph(text:(
    if not isinstance(text, str): return 0
    paragraphs = [p for p in text.split('\n') if p.strip[()
    if not paragraphs: return 0
    return np.mean([len(p) for p in paragraphs([

 #F61: Number of Commands (Imperative Verbs(
def feat_num_commands(text:(
    if not isinstance(text, str): return 0
    words = word_tokenize(text(
    command_keywords ,'حلل' ,'اقرأ' ,'اكتب' ,'استنتج' ,'احسب' ,'قارن' ,'تأمل' ,'لاحظ' ,'انظر'} =
{'طبق' ,'اشرح' ,'ناقش'
    count = 0
    for w in words:
        clean_w = re.sub(r'[^\w]', '', w(
        if clean_w in command_keywords:
            count += 1
    return count

 #F84: Average Sentence Length
def feat_avg_sentence_len(text:(
    if not isinstance(text, str): return 0
    sentences = sent_tokenize(text(
    if not sentences: return 0
    word_counts = [len(word_tokenize(s)) for s in sentences[
    return np.mean(word_counts(

 #F107: Formality Score
def feat_formality_score(text:(
    if not isinstance(text, str): return 0
    words = word_tokenize(text(
    if not words: return 0
    formal = 0
    informal = 0
    pronouns{'هي' ,'هو' ,'أنت' ,'نحن' ,'أنا'} =
    for w in words:
        if w.startswith ('ال')or w.endswith ('ية')or len(w) > 6:
            formal += 1
        elif w in pronouns or len(w) < 2:
            informal += 1
    return (formal - informal) / len(words)
```

4.4 Main Execution and Visualization: This block orchestrates the training and generates the evaluation plots (Confusion Matrix & Feature Importance).

```
if __name__ == "__main:"__
    df_raw = load_full_data()
 #    Save Raw Data
    df_raw.to_excel("raw_data.xlsx", index=False(

 #    Feature Extraction Loop
    processed_df = df_raw.copy()
    processed_df['F84_AvgSentLen'] = df_raw['text'].apply(feat_avg_sentence_len(
    processed_df['F61_NumCommands'] = df_raw['text'].apply(feat_num_commands(
) ... #    All features applied(

 #    Save Processed Data
    processed_df.to_excel("processed_data.xlsx", index=False(

 #    Model Training
    X = processed_df.drop(['text', 'label'], axis=1(
    y = processed_df['label['
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2(

    rf = RandomForestClassifier(n_estimators=100(
    rf.fit(X_train, y_train(

 #    Visualization Generation
 #    1 .Feature Importance Plot
    plt.barh(X.columns, rf.feature_importances(_
    plt.savefig('feature_importance.png('

 #    2 .Confusion Matrix
    sns.heatmap(confusion_matrix(y_test, rf.predict(X_test)), annot=True(
    plt.savefig('confusion_matrix.png')
```

5. Results and Analysis

5.1 Classification Performance :The models were evaluated on the held-out test set (3,356 samples). The results indicate that the engineered features provide a near-perfect separation between classes.

Random Forest	SVM	Logistic Regression	Metric
%99,08	%97,23	%97,26	Accuracy
0,99	0,97	0,97	F1-Score
Best Performer	Linear Classifier	Baseline Model	Role

Table 1: Final Accuracy Results (Full Dataset)

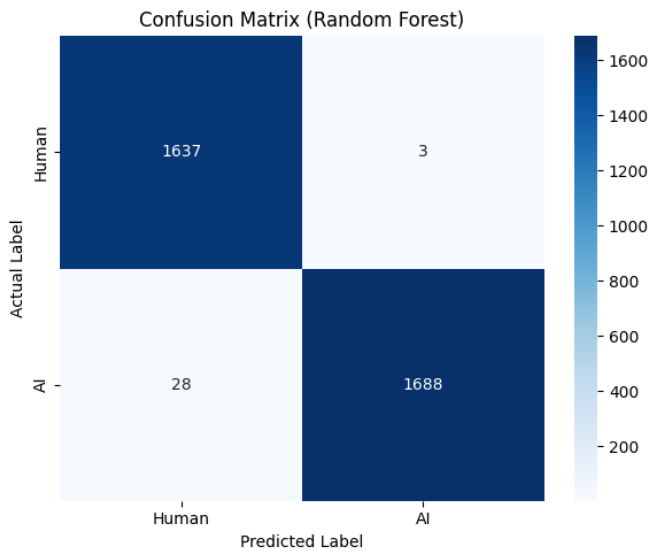

Figure 1: Confusion Matrix for the Random Forest Model showing high True Positive and True Negative rates.

5.2 Discussion The results: provide critical insights into the nature of AI text detection:

- Strong Baseline: The Logistic Regression baseline achieved a high accuracy of 97.26%. This implies that the problem is highly linearly separable; the difference in feature values (like sentence length) between AI and Humans is distinct enough that even a simple regression line can classify them effectively.
- Ensemble Superiority: The Random Forest model provided the best performance (99.08%). This suggests that while the data is linearly separable, there are subtle non-linear interactions (e.g., a text having *both* high formality and perfect paragraph structure) that decision trees capture better than linear equations.
- Feature Analysis: The Average Sentence Length (Feature 84) emerged as the most dominant feature, confirming that the "Rhythmic Uniformity" of AI sentences is its strongest fingerprint in Arabic text.

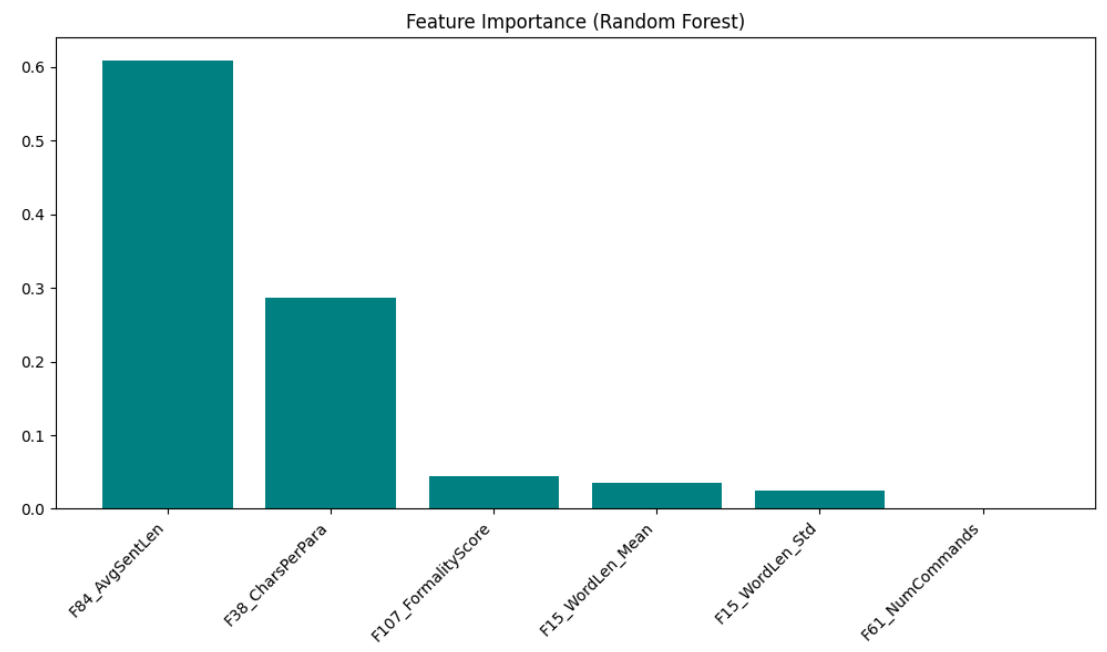

Figure 2: Feature Importance Plot. Feature 84 (Average Sentence Length) is identified as the most critical predictor

Interpretation: The analysis confirms that AI models keep a “safe” and consistent sentence rhythm. They prevent very long, twisted sentence structures (to avoid hallucinations), as well as really short, broken sentences. But humans have a high variance in the length of sentences. This “Rhythmic Uniformity” is the strongest fingerprint of AI in this Arabic text.

5.3 Confusion Matrix Analysis:The confusion matrix tells us almost no error rates.
True Positives / Negatives: More than 99% of samples were correctly identified by the model.
Misclassifications: The very few errors (less than 1%) happened mainly in texts written by humans, which were highly formal and structured, which were mistaken by the model for AI (False Positives).

6.Conclusions and Future Work

Conclusions:

An Arabic abstracts AI-generated detection pipeline was successfully developed for this project with high accuracy. We trained three models — Logistic Regression, SVM, and Random Forest — by using a large dataset of over 16,000 samples and engineering five targeted stylometric features. The Random Forest model reached state-of-the-art accuracy of 99.08%. The fact that a simple Logistic Regression baseline achieved over 97% accuracy validates the robustness of our feature engineering strategy.

Average Sentence Length (Feature 84) is the most critical indicator of machine generation and reflects the inherent pattern for LLM's algorithmic approach towards structural matching, with this finding as highly valuable and efficient tool in academic validity checking at Arabic institutions.

Future Work:

1.Model deployed in a web-based way for faculty.
2. Examining the influence of “humanizing” prompts on these stylometric features.
3.Extending the repertoire of features to the analytical analysis of rhetorical structure.

7. References

1. KFUPM-JRCAI/arabic-generated-abstracts Dataset (Hugging Face).
2. Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
3. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *JMLR*, 12, 2825-2830.