Project Log - Human vs AI Text Classification + Explainability

Notebook like (google colab):
https://colab.research.google.com/drive/1zh-8gsXeINEkCL7SfrLoDMjpw71cxBXb?usp=sharing

This project started with a fairly simple goal: distinguish human-written text from AI-generated text, and then understand *why* the model makes its decisions.

Over time it evolved into a much deeper exploration of style, explainability, and debugging ML pipelines.

I'm documenting everything here: dataset creation, training, mistakes, debugging, attribution, and insights.

I tried attempting it 2 times:

Attempt 1:

Dataset preparation:
Class 1:

I took 2 books in text format from Project Gutenberg.

1. The Interpretation Of Dream - Simund Freud
2. The Varieties of Religious Experience - William James

I cleaned the data in the following way:
A. The first thing I did was strip out the standard Project Gutenberg headers and footers. These are predictable and always marked by phrases like:

```
*** START OF THE PROJECT GUTENBERG EBOOK ***

*** END OF THE PROJECT GUTENBERG EBOOK ***
```

I wrote a function (strip_gutenberg_junk) that searches for these markers using regex and trims everything before the start marker and after the end marker. If the markers aren't found, the function safely defaults to keeping the whole text.

This step ensures that no licensing or distribution text leaks into the dataset.

B. Freud's book required special handling because of its heavy use of references, footnotes, and chapter metadata.

I explicitly chose a semantic start point for the book:

THE SCIENTIFIC LITERATURE ON THE PROBLEMS OF THE DREAM[D]

Everything before this - including the table of contents and prefaces - was discarded. This ensured that training text only came from actual prose, not structural material.

Then I removed several types of noise:

- Reference markers like [A23], [XIV], etc.
- Underscores (_) used to represent italics in plaintext
- Standalone Roman numerals that appeared on their own lines (typically chapter numbers or reference artifacts)
- Parenthetical citations such as (p. 42) or (pp. 103)

I also noticed that the book contained a large "LITERARY INDEX" section at the end, which I didn't want included. I searched for the phrase "LITERARY INDEX" and truncated the text there.

Finally, I normalized whitespace by collapsing all newlines, tabs, and multiple spaces into a single space, turning the entire book into a clean, continuous stream of prose.

All this was done using REGEX.

The cleaned output was saved as refined_dreams.txt.

C. William James's book had slightly different formatting issues, so I wrote a separate cleaning function.

Again, I picked a content-based starting point:

LECTURE I. RELIGION AND NEUROLOGY.

Everything before this - contents, prefaces, introductory material - was removed. Additional cleaning steps included:

- Removing underscores used for italics

- Removing long dashed separators (e.g. `----------`)
- Removing standalone Roman numerals
- Removing parenthetical page references
- Fixing hyphenated line breaks where words were split across lines (e.g. `experi- ence → experience`)

Like with Freud, I removed the ending index section by truncating the text at the `"INDEX"` marker. Finally, I collapsed all whitespace to ensure the text was a clean, uninterrupted sequence of words. This cleaned version was saved as `refined_religions.txt`.

D. Once both books were cleaned, I split them into smaller chunks suitable for model training.

I chose a chunk size of 150 words, stepping forward sequentially through the text. I only kept chunks longer than 100 words to avoid incomplete or low-information fragments. Each chunk was stored as a dictionary containing:

- the chunk text
- a human-readable label (`"Human"`)
- the author name (`"Freud"` or `"James"`)
- a class identifier

This produced two sets of human-authored training samples:

`class1_freud_chunks`

`class1_james_chunks`

Class 2 and 3:

I initially planned to use gemini flash model in the gemini api, but then realised it has very strict rate limits.

Then I shifted to Gemma, called by the api method, tried generating AI paragraphs, but on analyzing the output, realized that they were very similar, almost same I would say! I tried raising the temperature to the maximum (2), and also tried different parameter models of gemma, and penalizing parameter as well. But there were no significant changes in the paragraphs, read a reddit comment and realised that gemma model itself is deterministic in nature.

So I shifted to Qwen 3 8b model, now this one was creative.

For identifying core topics, I used tfidf to get significant words and tried to connect them, read the abstract of the books, and honestly ended up giving an llm the abstract of both the books with the significant words and told it to give 7 common topics relevant to both the authors, since authorship was used as the primary variable.

The Qwen model generated 72 paragraphs each for the topics, each having around 120 words.

The user prompt used was:
“Write a ~150-word paragraph about {topic}. Use a neutral, modern, objective AI tone. Avoid lists or bullet points. Single coherent paragraph only.”

(and I got trapped hare, which I realized after model training)

For class 3, I gave this user prompt:
“Write a ~150-word paragraph about {topic}. Use a neutral academic tone appropriate to this author. Avoid lists or bullet points. Single coherent paragraph only. Do not mention the author by name.”

With: "Freud": "Write in the analytical, clinical, introspective style of early psychoanalytic writing, focusing on unconscious processes, inner conflict, and symbolic meaning.",

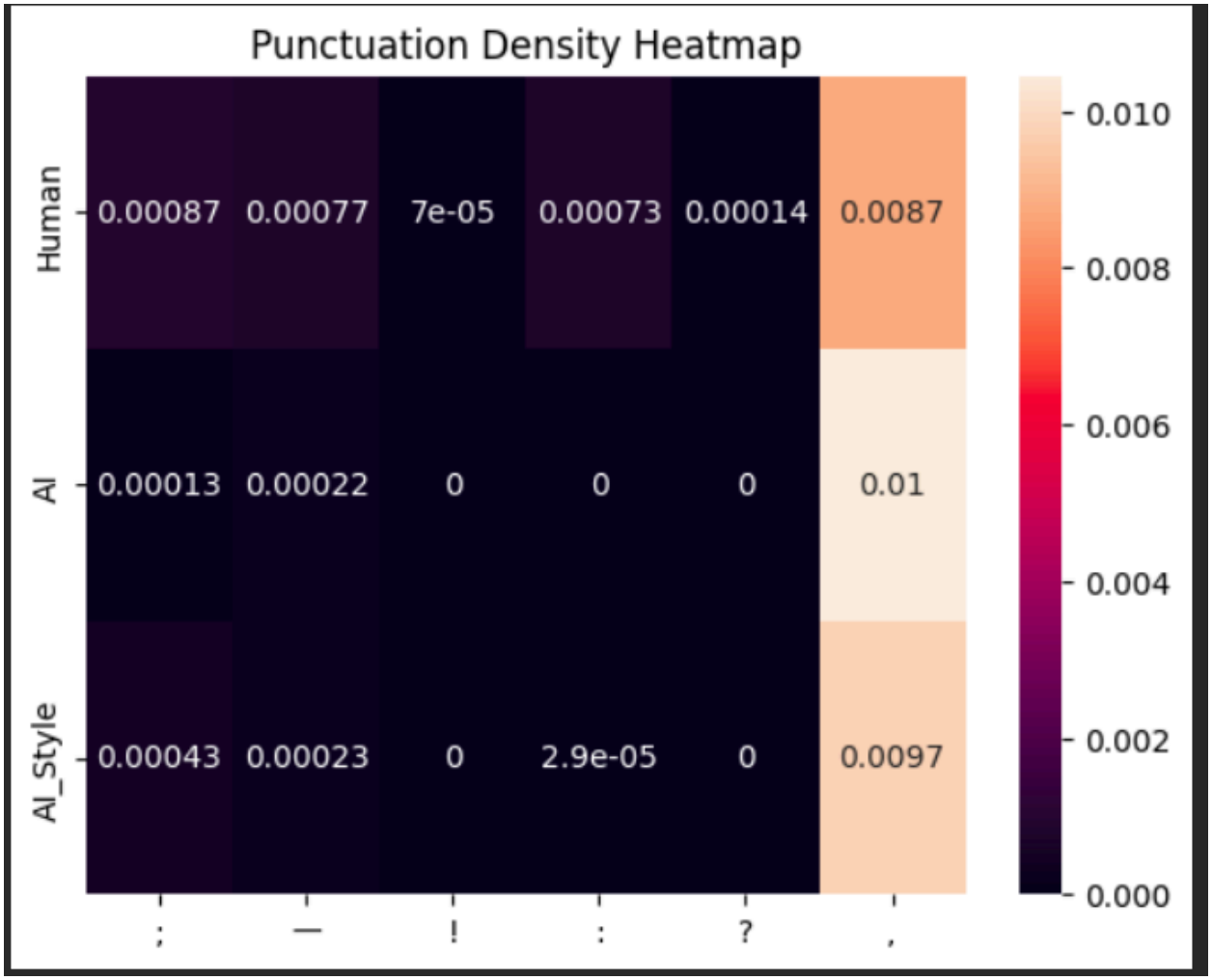
"William James": "Write in the reflective, philosophical, experiential style of William James, emphasizing lived experience, pragmatism, and personal meaning."

Stylistic/Lexical Analysis:

I did what the tasks said. For the 5000 word sample, I randomly picked up a 5000 contiguous word sample, and mainly used NLTK library.

	total_words	unique_words	TTR	hapax	adj_noun_ratio	avg_tree_depth	;	–	!	:	?	,	flesch_kincaid
Human	4948.0	1332.0	0.269200	806.0	0.320380	7.542105	0.000874	0.000769	0.00007	0.000734	0.00014	0.008742	12.392071
AI	4930.0	843.0	0.170994	409.0	0.528104	8.043902	0.000135	0.000216	0.00000	0.000000	0.00000	0.010447	19.147664
AI_Style	4981.0	1033.0	0.207388	528.0	0.436170	8.005000	0.000429	0.000229	0.00000	0.000029	0.00000	0.009727	17.090813

The stylistic statistics showed a clear separation between human and AI-generated text. Human writing exhibited higher lexical diversity (higher TTR and hapax counts), suggesting richer and more varied vocabulary use. In contrast, both AI and AI-style text relied more heavily on adjectival - nominal constructions and displayed greater syntactic depth, indicating structurally complex but lexically constrained prose. Additionally, AI-generated text scored substantially higher on readability metrics, reflecting longer sentences and denser academic-style phrasing.



The punctuation heatmap showed that human text uses a wider variety of punctuation marks (notably semicolons, em dashes, and colons), while AI and AI-style text relied almost exclusively on commas. Exclamation marks and question marks were virtually absent in AI outputs, indicating flatter expressive range. Overall, AI writing exhibited reduced punctuation diversity but heavier comma usage, reinforcing its tendency toward long, structurally dense sentences rather than nuanced rhetorical variation.

	Mean Tree Depth	Std Dev Tree Depth
Human	7.542105	2.776938
AI	8.043902	2.086665
AI_Style	8.005000	1.845257

The tree depth showed that AI used more nested phrases/sentences (by mean), while Humans had a high standard dev, meaning the variance in their paragraphs was higher.

Building the Detectors

For each model, I combined all classes in a concatenated data frame. Then I divided the data frame in train and test 80:20 ratio. The paragraphs were randomly chosen and put in the divisions.

Tier A:

The random forest classifier gave these results on the test set:

...	precision	recall	f1-score	support
0	0.97	0.99	0.98	462
1	0.82	0.65	0.73	101
2	0.65	0.75	0.70	91

It performed well on detecting human generated paragraphs possibly because of how the metrics were very different for human (punctuation, ttr, etc.), but had 70% accuracy in detecting AI generated text.

Tier B:

I used GloVe embeddings by importing gensim library.

I started by combining all text sources into a single dataset. I appended cleaned Freud and William James chunks as label 0 (human), followed by two AI datasets labeled 1 and 2 respectively. This gave me a unified corpus with three classes.

Next, I computed TF-IDF vectors over the entire dataset using a vocabulary capped at 5000 terms. This step was meant to capture statistically important words across human and AI writing. I then used a custom `embed_tfidf()` function to convert each paragraph into a fixed-length numerical embedding (300 dimensions), producing a feature matrix `X`. Labels were stored separately as `y`.

At this point, the pipeline followed a classic machine-learning structure:

`Text → TF-IDF → Dense embedding → Neural classifier`

I split the data into training and test sets (80/20) using `train_test_split`, then converted everything into PyTorch tensors.

To classify these embeddings, I implemented a simple feed-forward neural network with three layers:

- Input layer (300 → 256)
- Hidden layer (256 → 64)
- Output layer (64 → 3)

ReLU activations were used throughout, along with dropout for regularization. The model was trained using Adam optimization and cross-entropy loss for 30 epochs.

After training, I evaluated performance on the held-out test set and printed a standard classification report:

```
...              precision    recall  f1-score   support

           0       0.91      1.00      0.95       454
           1       0.60      0.95      0.74        98
           2       1.00      0.01      0.02       102
```

This stage essentially served as my **Tier-B detector**: a lightweight neural baseline operating purely on engineered features rather than raw text. The goal here was not state-of-the-art performance, but to establish whether lexical and statistical representations alone could separate human and AI writing.

As seen, the three classes were not distinguished reasonably well. Tier B relied on shallow, hand-engineered lexical features, which captured surface statistics but failed to model higher-order discourse structure present in AI-generated text. As a result, it struggled to generalize beyond obvious stylistic cues and could not reliably distinguish nuanced AI paragraphs from human prose.

In summary, this phase acted as an exploratory baseline: I first tested whether classical embeddings plus a feed-forward network could detect authorship, before moving on to DistilBERT and Integrated Gradients for more expressive modeling and interpretability.

Tier C:

Tier C was implemented by fine-tuning a pretrained transformer model using LoRA (Low-Rank Adaptation) instead of updating all model parameters. I wrapped the base sequence classification model with LoRA adapters, which inject small trainable low-rank matrices into the attention layers while keeping the original weights frozen. This allowed me to adapt the model to the three-class authorship task (Human, AI_1, AI_2) using only a fraction of the parameters required for full fine-tuning, making training feasible on a 4GB RTX 3050.

The model was trained directly on raw paragraph text using the same dataset splits as earlier stages. Inputs were tokenized normally and passed through the frozen transformer backbone, with only the LoRA layers and classification head being optimized using cross-entropy loss. Because the number of trainable parameters was small, training was stable and fast, and GPU memory usage remained low throughout.

After training, predictions were generated on the held-out test set, and performance was evaluated using standard classification metrics. The LoRA-based detector was then integrated with the explainability pipeline by applying Integrated Gradients at the embedding layer, allowing token-level attribution analysis without modifying the underlying architecture. This made Tier C both computationally efficient and compatible with interpretability, serving as the final detection stage in the system.

These were the results (which were interesting)

	precision	recall	f1-score	support
0	1.00	1.00	1.00	430
1	0.85	0.92	0.88	112
2	0.91	0.84	0.87	112
accuracy			0.96	654
macro avg	0.92	0.92	0.92	654
weighted avg	0.96	0.96	0.96	654

An F1 score for human vs AI! I analyzed, if there was any topic leakage in the dataset, but no such things came up. The reason this could have happened was because I used this in my AI generated class 1 prompt: Use objective and neutral AI tone.

So it could be possible that my model learnt the difference between the neutral AI tone, and the 1900-ism english that was used in the book. Yes, the books were academic, and used somewhat archaic english. This gave the model a lot of hints.

TASK 3: What does the model picks up as AI-isms?

```
=== AI ===
collectively 0.374
situations   0.335
between      -0.319
circumstances 0.284
beings       0.283
allowing     0.268
enable       0.258
world        0.257
actively     0.254
in           -0.241
under        0.235
,            0.234
,            0.230
.            0.229
characterized 0.218
```

The output suggest how the model let some fancy words decide its prediction.

Task 4

I ran the model with my SOP (which I 100% wrote on my own). The model said AI. And then when I gave it the same thing but rephrase in the manner of 1900s authors + academic styles, the result was…

Human.

This proves that the model took on the writing style of the authors chose (1900s + academic) and judged anything incorporating this style as human.

End of the report.