Appendix

A RESULTS SUMMARY FOR HUMAN RIGHTS VIOLATION

For gemini (Intra-LLM setup), PerRR_GP_G and PreRR_GP_G are observed to be 92.011 and 0.9613, respectively, while for llama3 (Intra-LLM setup), these values are recorded as 95.859 and 0.9609.

While the Intra-LLM metrics for gemini align closely with those presented in Table 2 , the llama3 configuration demonstrates superior performance in the Intra-LLM setup across the entire dataset comprising 2,377 cases. In the Inter-LLM setup, specifically for the gemini → llama3 configuration, the PerRR_GP_L, PreRR_GP_L, PerRR_GA_L, and PreRR_GA_L values are 93.453, 0.9377, 98.433, and 0.9579, respectively.

Conversely, the corresponding values for the llama3 → gemini setup, specifically PerRR_LP_G, PreRR_LP_G, PerRR_LA_G, and PreRR_LA_G, are 95.396, 0.9550, 91.602, and 0.9327, respectively. Notably, for the larger dataset, the results are either comparable or demonstrate improvement, particularly in the case of PerRR_LA_G for the llama3 → gemini configuration.

To evaluate LLM agnosticism, we assess whether algorithms generated by one LLM generalize to another. [55] shows that state-of-the-art LLMs can effectively execute natural language-described algorithms. By testing the same algorithm across two LLMs, we can measure their procedural consistency.

B RESULTS FOR FINANCIAL TASKS

In the intra-LLM setup for Apple (AAPL), we observed the performance of gemini and llama3-70b models using ReQuest algorithm $\mathcal{A}$. The baseline Macro F1 score for BERTweet was 0.53. When gemini was employed, the initial Macro F1 score was 0.44, which improved to 0.48 after applying $\mathcal{A}$. This improvement in accuracy was reflected in the high reproducibility percentage (PerRR_GP_G) of 91%. On the other hand, llama3-70b started with a Macro F1 score of 0.41 and showed a slight decrease to 0.38 with ($\mathcal{A}$) resulting in a PerRR_LP_L of 92.68%, indicating high reproducibility but a drop in performance. In the Inter-LLM setup, we examined the cross-model reproducibility where ReQuest algorithm $\mathcal{A}$ generated by gemini was applied to llama3-70b and vice versa. When ReQuest algorithm $\mathcal{A}$ from gemini was applied to llama3-70b, the PerRR_GP_L was 91.67%, suggesting a high degree of reproducibility, though the Macro F1 score was moderately lower at 0.48. Conversely, applying ReQuest algorithm $\mathcal{A}$ from llama3-70b to gemini resulted in a PerRR_LP_G of 78.84%, indicating a lower reproducibility and a Macro F1 score of 0.52.

The evaluations for Google (GOOG) followed a similar pattern. The baseline Macro F1 score for BERTweet was 0.55. For gemini, the initial Macro F1 score was 0.45, which slightly improved to 0.46 after applying ReQuest algorithm $\mathcal{A}$ with a PerRR_GP_G of 97.82%, showing high reproducibility. llama3's initial Macro F1 score of 0.37 decreased to 0.35 with $\mathcal{A}$, achieving a PerRR_LP_L of 94.59%. In the Inter-LLM setup for Google, the cross-model reproducibility was again assessed. When ReQuest algorithm $\mathcal{A}$ from gemini was applied to llama3, the PerRR_GP_L was 93.75%, and the Macro F1 score was 0.48. Applying $\mathcal{A}$ from llama3 to gemini resulted in a PerRR_LP_G of 71.15%, with a lower Macro F1 score of 0.52.

Results for two new datasets have been added in Table 33, where the stock movement for Microsoft and Amazon were predicted. Similar trends were observed for the Microsoft (MSFT) and Amazon (AMZN) datasets. For the Microsoft dataset, the baseline Macro F1 score with BERTweet was 0.51. Gemini's performance improved from 0.42 to 0.51 with ReQuest, achieving a PerRR_GP_G of 82%. Upon employing llama3, the Macro F1 score declined from 0.41 to 0.37, despite a high PerRR_LP_L of 90.2%. In the inter-LLM setup, gemini's ReQuest applied to llama3 achieved a high PerRR_GP_L of 95.45%, while llama3 to gemini showed a PerRR_LP_G of 91.11%. For the Amazon dataset, gemini exhibited an increase in performance with ReQuest, improving from 0.47 to 0.49, whereas llama3 demonstrated a decline from 0.43 to 0.37. The inter-LLM reproducibility for the Amazon dataset was observed to be lower than that for the Microsoft dataset, with PerRR_GP_L recorded at 82.97% and PerRR_LP_G at 87.75%.

C RESULTS SUMMARY FOR HEALTH TASKS

In T_1, the **PreRR_GP_G** score of 0.617 indicates that the algorithm generated by gemini-1.0-pro is able to replicate a decent proportion of its initial predictions at the data point level, as is the case with llama3 with a **PreRR_LP_L** score of 0.701. But for T_2, low scores of both **PreRR_GP_G** and **PreRR_LP_L** indicate the reasoning process of LLMs may be inconsistent and highly sensitive to the prompt structure and wording.

For T_1, in the "inter-LLM" setup, when gemini's algorithm, $\mathcal{A}$ is executed on llama3 (gemini → llama3), the **PerRR_GP_L** score of 95.47% conveys very consistent performance when compared to gemini's task prompt responses, which suggests the reasoning processes of the two LLMs are in parity. Further, the green arrow reveals llama3's responses using gemini's $\mathcal{A}$ are better than gemini's task prompt responses. A **PreRR_GP_L** score of 0.728 indicates a healthy number of exact matches, which, though ever so slightly, increases to 0.731 when gemini's $\mathcal{A}$ is 'executed' on both gemini and llama3 (**PreRR_GA_L**). A low score of (**PerRR_GA_L**) 86.47% indicates llama3's reasoning is not analogous to gemini's $\mathcal{A}$ even though the former responds better with gemini's $\mathcal{A}$. In case of llama3 → gemini, a near perfect score of **PerRR_LP_G** (99.31%) may indicate a striking similarity in llama3's internal mechanism and it's $\mathcal{A}$. Further, the high value of **PerRR_LA_G** (94.99%) accounts for the overlap between results of llama3's algorithm, $\mathcal{A}$ being executed on llama3 and gemini. However, this does not take away the fact that their predictions are not in correspondence with the true labels in the dataset.

For T_2, while gemini's **PerRR_GP_G** is higher (97.545%), the **PreRR_GP_G** of 0.672 is still low, suggesting that individual predictions are not reproducible in totality. llama3, however, shows a significant incredible performance in both **PerRR_LP_L** and **PreRR_LP_L**, implying that its algorithms are very effective in replicating its original performance, especially at the individual prediction level.

Unlike T_1, $\mathcal{A}$ produced by both `gemini` and `llama3` performs poorly for the inter-LLM setup in T_2.

D WHY USE PRERR WHEN WE ALREADY HAVE MACRO-F1 BASED PERRR?

A scenario in which the PerRR score may be misleading is when the macro f1 score of any 2 prediction sets to be evaluated is low especially for a relatively simple task like binary classification where the odds of being right are 50% for each data point. In such cases there might be more than one unique prediction set to yield the same low macro-f1 score. This will result in multiple unique and pred_sets pairs where the PerRR is 100%. An example of such case is given below:

Suppose we have a binary classification problem on a dataset of size 6. The gold standard labels are given by *gold_std* = {1,0,1,0,1,0} i.e it is a perfectly balanced dataset. Lets say LLM1 and LLM2 predicted *pred_set_1* = {0,0,0,0,1,1} and *pred_set_2* = {0,0,0,1,1,0} respectively. So Macro_F1(*pred_set_1*) = Macro_F1(*pred_set_2*) = 0.4857. Hence PerRR(*pred_set_1*,*pred_set_2*) = 100%. But *pred_set_1* and *pred_set_2* are clearly different, leading to a misleading sense of reproducibility if one was to only rely on PerRR. In fact in this particular example there are 18 such *pred_sets* which will have 100% PerRR with *pred_set_1*. This is actually the peak of the distribution given in Figure 6. This calls for necessity to introduce a stricter measure of similarity for such cases, which is fulfilled by the PreRR metric. Here the PreRR score will be 0.66 which accurately depicts the difference between both pred_sets.

If the complexity of the task is increased to a 3-class multi-class classification problem on same size dataset, most of the distribution shifts towards left as shown in Figure 7 where the *gold_std* = {0,1,2,0,1,2}. Following the same behaviour for even more complex task like multi-label classification, the distribution should shift even more towards the left. This distribution directly signifies the probability of discrepancy between the PerRR and PreRR score for a given classification problem. As most of the distribution is located in the left half of the Macro-F1 axis, the robustness of PerRR score decreases for low macro-F1 score for simple problems like binary classification and very low macro-F1 score for more complex tasks like multi-label classification.

Due to inherent strict nature of PreRR, its variance should increase as the task complexity increases. Simpler tasks like binary classification should have little variations between pred_sets. For more complex tasks like multi-label classification, it should vary more relative to PerRR for slightest of mistakes (especially at lower macro-F1s).

Definition of some variables {training_prompt}:

Statute ID: Indian Penal Code, 1860_147 Title: Punishment for rioting Description: Whoever is guilty of rioting, shall be punished with imprisonment of either description for a term which may extend to two years, or with fine, or with both.

###

Statute ID: Indian Penal Code, 1860_149

Title: Every member of unlawful assembly guilty of offence committed in prosecution of common object

Description: If an offence is committed by any member of an unlawful assembly in prosecution of the common object of mat assembly, or such as the members of that assembly knew to be likely to be committed in prosecution of that object, every person who, at the time of the committing of that offence, is a member of the same assembly, is guilty of that offence.

............(for 18 statutes)

###

Task:

You are given a fact statement delimited by triple backticks and statutes with their title and description. Your task is to identify the statutes applicable to the fact statement from the given statutes that you are most confident apply to the fact statement. Each statute consists of a title and a description of its scope and provisions. Include only those statute in your response which description logically matches with some parts of the fact statement. Training:

Statutes:

{training_prompt}

Fact Statement:

“‘{inp}”’

Response and Instructions:

Format of response: Statute1; Statute2 ...

Your response should include the statutes applicable to the fact statement. The applicable statute must be mentioned exactly as it appears in Statutes provided. Include only those statutes which you are very sure about.

Table 6: Task Prompt: Statute Prediction (Gemini); inspired by [42]

Task:

You are given a fact statement delimited by triple backticks and statutes with their title and description. Your task is to identify the statutes applicable to the fact statement from the given statutes that you are most confident apply to the fact statement. Each statute consists of a title and a description of its scope and provisions. Include only those statute in your response which description logically matches with some parts of the fact statement.

Training:

Statutes:

{training_prompt}

Fact Statement:

“‘{inp}“‘

Your response should include the statutes applicable to the fact statement. The applicable statute must be mentioned exactly as it appears in Statutes provided. Include only those statutes which you are very sure about. No need to provide any explanation. Return output in the format of response. No need to write ’Here is the response: ’

Format of response: Statute1; Statute2 …

Table 7: Task Prompt: Statute Prediction (Llama3-70b); inspired by [42]

1. **Read and understand the fact statement.** Identify the key legal issues in the fact statement.
2. **Identify the relevant area of law.** For example, if the fact statement involves a criminal offense, then the relevant area of law is criminal law.
3. **Research the relevant statutes.** This may involve using a legal database or consulting with a legal professional.
4. **Analyze the statutes and determine which ones are applicable to the fact statement.** Consider the following factors:
* The elements of the offense
* The defenses to the offense
* The penalties for the offense
5. **Determine the strength of the evidence.** Consider the following factors:
* The strength of the prosecution's case
* The strength of the defense's case
* The likelihood of a conviction

Example:

Fact statement: A person is charged with murder.

Steps to identify applicable statutes:

1. **Read and understand the fact statement.** The key legal issue in the fact statement is murder.
2. **Identify the relevant area of law.** The relevant area of law is criminal law.
3. **Research the relevant statutes.** The relevant statute is the Indian Penal Code, 1860_302, which defines murder as "whoever commits murder shall be punished with death, or imprisonment for life, and shall also be liable to fine."
4. **Analyze the statute and determine if it is applicable to the fact statement.** The statute is applicable to the fact statement because the person is charged with murder.
5. **Determine the strength of the evidence.** This step is not necessary for the purpose of identifying applicable statutes. However, it may be relevant for other purposes, such as determining the likelihood of a conviction.

Table 8: ReQuest Algorithm: Statute Prediction (Gemini)

Step 1: Parse the Fact Statement
* Break down the fact statement into individual events, actions, and circumstances.
* Identify the key entities involved, such as people, objects, and locations.

Step 2: Extract Relevant Information
* Extract relevant information from the fact statement, including:
+ Actions taken by individuals (e.g., shooting, assaulting, surrendering)
+ Consequences of actions (e.g., injury, death)
+ Objects involved (e.g., weapons, vehicles)
+ Locations and settings (e.g., village, police station)

Step 3: Match with Statute Descriptions
* Compare the extracted information with the descriptions of the statutes provided.
* Look for statutes that mention similar actions, consequences, objects, or settings.

Step 4: Filter and Rank Statutes
* Filter out statutes that do not match the extracted information.
* Rank the remaining statutes based on their relevance and similarity to the fact statement.

Step 5: Select Applicable Statutes
* Select the top-ranked statutes that are most relevant to the fact statement.
* Ensure that the selected statutes are applicable to the events, actions, and circumstances described in the fact statement.

Step 6: Return the Response
* Return the applicable statutes in the required format, separated by semicolons.

Table 9: ReQuest Algorithm: Statute Prediction (Llama3-70b)

```
Task:
You are a bot who strictly follows steps. You strictly follow algorithms with very deterministic output. You are given a fact statement delimited by triple backticks (“‘) and statutes with their title and description. Your task is to follow the given steps to find out the statutes applicable. Each statute consists of a title and a description of its scope and provisions.
Statutes:
{training_prompt}
##
Steps to follow:
{algo}
##
Fact Statement:
“‘{inp}“‘ Response and Instructions:
Format of response: Statute1; Statute2 ...
The applicable statutes must be mentioned exactly as it appears in Statutes provided.
Include only those statutes which you get with very high confidence from the algorithm. STRICTLY USE THE SPECIFIED STEPS FOR PREDICTION. DO NOT USE ANY OTHER ALGORITHM.
```

Table 10: Robustness Check Prompt: Statute Prediction (Gemini). The algorithm is in Table 8.

```
Task:
You are a bot who strictly follow steps. You strictly follow algorithms with very deterministic output. You are given a fact statement delimited by triple backticks (“‘) and statutes with their title and description. Your task is to follow the given steps to find out the statutes applicable. Each statute consists of a title and a description of its scope and provisions.
Statutes:
{training_prompt}
##
Steps to follow:
{algo}
##
Fact Statement:
“‘{inp}“‘
What relevant statutes did you get after following these steps for this fact statement? Return them in the following response format:
Format of response: Statute1; Statute2 ...
Your response should be such that I can extract the statutes using str.split(’;’). Do not return anything else.
```

Table 11: Robustness Check Prompt: Statute Prediction (Llama3-70b). The algorithm is in Table 9.

```
import re

def identify_applicable_statutes(fact_statement):
     Identify the key facts in the fact statement.
    key_facts = []
    for line in fact_statement.split(""):
        line = line.strip()
        if line:
            key_facts.append(line)
    # Determine which statutes may be applicable to the facts.
    potentially_applicable_statutes = []
    for statute in statutes:
        for key_fact in key_facts:
            if re.search(key_fact, statute["description"]):
                potentially_applicable_statutes.append(statute)
    # Analyze the statutes to determine if they are actually applicable to the facts.
    applicable_statutes = []
    for statute in potentially_applicable_statutes:
        if is_statute_applicable(fact_statement, statute):
            applicable_statutes.append(statute)
     Return the list of applicable statutes.
    return applicable_statutes

def is_statute_applicable(fact_statement, statute):
    # Check if the fact statement contains any of the key terms from the statute’s description.
    for key_term in statute["description"].split(" "):
        if re.search(key_term, fact_statement):
            return True
    # Check if the fact statement describes a situation that is covered by the statute.
    for example in statute["examples"]:
        if re.search(example, fact_statement):
            return True
    # Otherwise, the statute is not applicable.
    return False
```

Table 12: Python code generated by Gemini for Statute Prediction Task

```
import re

def identify_applicable_statutes(fact_statement):
    key_facts = []
    for line in fact_statement.split(""):
        line = line.strip()
        if line:
            key_facts.append(line)
    potentially_applicable_statutes = []
    for statute in statutes:
        for key_fact in key_facts:
            if re.search(key_fact, statute["description"]):
                potentially_applicable_statutes.append(statute)
    applicable_statutes = []
    for statute in potentially_applicable_statutes:
        if is_statute_applicable(fact_statement, statute):
            applicable_statutes.append(statute)
    return applicable_statutes

def is_statute_applicable(fact_statement, statute):
    for key_term in statute["description"].split(" "):
        if re.search(key_term, fact_statement):
            return True
    for example in statute["examples"]:
        if re.search(example, fact_statement):
            return True
    return False
```

Table 13: Python code (comments removed) generated by Gemini for Statute Prediction Task

	w.r.t. Task Prompt		w.r.t. ReQuest Algo	
Python Code	**Per_RR**	**Pre_RR**	**Per_RR**	**Pre_RR**
With comments	100	0.612	98.15	0.6643
Without comments	97.42	0.6063	99.33	0.6094

Table 14: Results obtained after "executing" Python code generated by Gemini by prompting (Table 10)

The PerRR score of both the versions of the Python script (with comments and without comments) prediction (by ‘executing’ it within Gemini itself) with respect to task prompt (Table 6) is very high. This shows that the performance of both the versions of the Python codes are similar with respect to the ReQuest algorithm and the task prompt. The PreRR score of both the versions is higher than the PreRR score of ReQuest algorithm predictions with respect to task prompt predictions. The Macro-F1 score of the Python script (with comments) ‘executed’ on Gemini by prompting is equal to the Macro-F1 score of the predictions generated by the task prompt on the same. However the relatively lower PreRR score suggests that the predictions were not exactly replicated by the Python script despite resulting in the same Macro-F1 score. It was also observed that removing the comments from the python script (which would have given additional information to the LLM in form of natural language) decreased the PreRR score both with respect to the task prompt and ReQuest algorithm.

Table 15: Insights gained from Table 14

You are given a court statement delimited by triple backticks. Your task is to identify whether any
human rights violation occurred by studying the court statement. Refer to the website link given below for
relevant Articles and Protocols. Use only those articles and protocols mentioned in the website.
Website link: http://www.hri.org/docs/ECHR50.html
Court statement: “‘{query}”’
Response format: 1 if any one of given article or protocol is violated and 0 if none of them are violated.
If some articles or protocols are violated, mention them at the end with relevant sentences from the statement that led to the violation.
NOTE: THERE CAN BE MULTIPLE VIOLATIONS IN A SINGLE STATEMENT. ONLY RETURN THOSE ARTICLES WHICH YOU ARE MOST CONFIDENT ABOUT ELSE RETURN 0.

Table 16: Task Prompt: Human Rights Violation Prediction (Gemini)

You are given a court statement delimited by triple backticks. Your task is to identify whether any human rights violation occured by studying the court statement. Refer to the ECHR website link given below for relevant Articles and Protocols. Use only those articles and protocols mentioned in the website.
Website link: http://www.hri.org/docs/ECHR50.html
Court statement: “‘{query}“‘
If some articles or protocols according to ECHR are violated, mention them at the end with relevant sentences from the statement that led to the violation. NOTE: THERE CAN BE MULTIPLE VIOLATIONS OR NO VIOLATIONS IN A SINGLE STATEMENT. ONLY RETURN THOSE ARTICLES WHICH YOU ARE MOST CONFIDENT ABOUT ELSE RETURN 0.
Response format: 1 if any one of given article or protocol is violated and 0 if none of them are violated. The 1 (if any article or protocol is violated) or 0 (if no article or protocol is violated) should be on the first line followed by a newline character.

Table 17: Task Prompt: Human Rights Violation Prediction (Llama3-70b)

1. **Identify the relevant facts in the statement:**
- Who is the applicant?
- What are the applicant's claims?
- What are the facts of the case?
- What was the outcome of the case?

2. **Identify the relevant articles in the human rights treaty:**
- Which human rights treaty is applicable to the case?
- Which articles of the treaty are relevant to the applicant's claims?

3. **Analyze the facts in light of the relevant articles:**
- Do the facts of the case violate any of the articles of the treaty?
- If so, which articles are violated?

Example:

Court statement:

> The applicant is a journalist who was arrested and detained for three days without charge. He was not allowed to contact a lawyer or his family during his detention.

Steps to identify human rights violations:

1. **Identify the relevant facts in the statement:**
- The applicant is a journalist.
- The applicant was arrested and detained for three days without charge.
- The applicant was not allowed to contact a lawyer or his family during his detention.

2. **Identify the relevant articles in the human rights treaty:**
- The Universal Declaration of Human Rights (UDHR) is applicable to the case.
- Articles 9, 10, and 11 of the UDHR are relevant to the applicant's claims.

3. **Analyze the facts in light of the relevant articles:**
- Article 9 of the UDHR protects the right to liberty and security of person. The applicant's detention without charge violated his right to liberty and security of person.
- Article 10 of the UDHR protects the right to freedom of expression. The applicant's arrest and detention for his journalistic activities violated his right to freedom of expression.
- Article 11 of the UDHR protects the right to be treated with humanity and respect for the inherent dignity of the human person. The applicant's treatment during his detention, including being denied access to a lawyer and his family, violated his right to be treated with humanity and respect for his inherent dignity.

Table 18: ReQuest Algorithm: Human Rights Violation Prediction (Gemini)

Step 1: Read and understand the court statement
* Read the entire court statement to understand the events, allegations, and circumstances described.
* Take note of key events, dates, times, locations, and individuals involved.
Step 2: Identify potential human rights issues
* Look for keywords and phrases that may indicate human rights violations, such as:
+ Physical harm or abuse
+ Coercion or intimidation
+ Discrimination or bias
+ Unfair treatment or procedure
+ Violations of privacy or dignity
* Identify specific incidents or events that may constitute human rights violations.
Step 3: Consult relevant human rights frameworks
* Refer to relevant human rights frameworks, such as the European Convention on Human Rights (ECHR), the Universal Declaration of Human Rights (UDHR), or other applicable national or international human rights instruments.
* Familiarize yourself with the articles, protocols, and provisions that protect human rights.
Step 4: Match allegations with human rights frameworks
* Compare the allegations and incidents identified in Step 2 with the relevant human rights frameworks consulted in Step 3.
* Determine whether the allegations, if true, would constitute a violation of specific human rights provisions.
Step 5: Analyze evidence and context
* Evaluate the evidence presented in the court statement, including witness statements, documents, and other evidence.
* Consider the context in which the alleged violations occurred, including the circumstances, motivations, and intentions of the individuals involved.
Step 6: Determine human rights violations
* Based on the analysis in Step 5, determine whether the allegations constitute a violation of human rights provisions.
* Identify the specific articles, protocols, or provisions that are violated, and describe the nature of the violation.
Step 7: Document findings
* Clearly document the findings, including the specific human rights violations identified, the relevant provisions violated, and the evidence supporting the conclusion.
* Provide a concise and objective summary of the human rights violations identified in the court statement.

Table 19: ReQuest Algorithm: Human Rights Violation Prediction (Llama3-70b)

Task:
You are a bot who strictly follow steps. You strictly follow algorithms with very deterministic output. You are given a court statement delimited by triple backticks (“‘). Your task is to follow the given steps to find out if any human right article or protocol is violated. Refer to the website link given below for
relevant Articles and Protocols: ONLY USE THOSE PROTOCOLS AND ARTICLES MENTIONED IN THE WEBSITE BELOW.
Website link: http://www.hri.org/docs/ECHR50.html
Steps to follow:
{algo}
##
Court statement: “‘query”’
##
Format of response: 1 if any one of given article or protocol is violated and 0 if none of them are violated. Anyting else at the end.
STRICTLY USE THE SPECIFIED STEPS FOR PREDICTION. DO NOT USE ANY OTHER ALGORITHM. If some articles or protocols are violated, mention them at the end with the relevant steps provided by me that led you to that conclusion.
NOTE: THERE CAN BE MULTIPLE VIOLATIONS IN A SINGLE STATEMENT. ONLY RETURN THOSE ARTICLES (from ECHR) WHICH YOU ARE MOST CONFIDENT ABOUT ELSE RETURN 0.

Table 20: Robustness Check Prompt: Human Rights Violation (Gemini). The algorithm is in Table 18.

Task:
You are a bot who strictly follow steps. You strictly follow algorithms with very deterministic output. You are given a court statement delimited by triple backticks (“‘). Your task is to follow the given steps to find out if any articles or protocols according to ECHR are violated. Refer to the ECHR website link given below for relevant Articles and Protocols. Use only those articles and protocols mentioned in the website.
Website link: http://www.hri.org/docs/ECHR50.html
Steps to follow:
{algo}

Court Statement:
“‘{inp}“‘
STRICTLY USE THE SPECIFIED STEPS FOR PREDICTION. DO NOT USE ANY OTHER ALGORITHM. If some articles or protocols are violated, mention them at the end with the relevant results from steps provided by me that led you to that conclusion.
Format of response: 1if any one of given article or protocol is violated and 0if none of them are violated.
The 1 (if any article or protocol is violated) or 0 (if no article or protocol is violated) should be on the first line followed by a newline character.
NOTE: THERE CAN BE MULTIPLE VIOLATIONS IN A SINGLE STATEMENT OR NONE. Return 0 if you are even slightly unsure if any right is violated as we dont want false convictions.

Table 21: Robustness Check Prompt: Human Rights Violation (Llama3-70b). The algorithm is in Table 19.

You are a highly skilled financial analyst with expertise in predicting stock price movements based on sentiment analysis. Your task is to determine whether the stock price of a given ticker will rise or fall based on the sentiment expressed in tweets related to that ticker.
Instructions:
1. Analyze the provided tweets to assess the overall sentiment.
2. If the overall sentiment is positive, predict that the stock price will rise (output 1).
3. If the overall sentiment is negative, predict that the stock price will fall (output 0).
Note 1: If you find any negative sentiment in the tweets, focus more on that.
Most of the tweets are positive or neutral, so it is crucial to pay close attention to negative sentences. Negative sentiment carries more weight than positive sentiment in this analysis, so prioritize identifying and considering negative expressions but don’t forget about positive ones.

Table 22: Task Prompt: Stock Prediction

Robust Sentiment Analysis Algorithm
Input: A set of tweets related to a given stock ticker.
Output: The overall sentiment of the tweets, represented by a numerical score.
Steps:
1. **Preprocess the tweets:** Remove irrelevant tweets and tokenize/lowercase the text of each tweet. 2. **Use a sentiment analysis tool to assign a sentiment score to each tweet:** I used the VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis tool, which is specifically designed to analyze social media text. VADER assigns a compound sentiment score to each tweet, ranging from -1 (negative) to +1 (positive). VADER considers the lexical features of the text, as well as the presence of negations and intensifiers, to determine the sentiment.
3. **Calculate the overall sentiment:** Take a weighted average of the sentiment scores of all the tweets, where recent tweets are given more weight. I used a simple linear weighting scheme, where the weight of a tweet is inversely proportional to its age. 4. **Apply robustness measures:** To improve the robustness of the algorithm, I applied the following measures:
* **Negation handling:** I identified and handled negations in the tweets to ensure that negative sentiment was correctly captured. For example, the tweet “AAPL is not a good investment” would be assigned a negative sentiment score, even though it contains the word “not.”
* **Contextual analysis:** I used VADER’s ability to consider the context and tone of the tweets to improve the accuracy of the sentiment analysis. For example, the tweet “AAPL is a great company, but their stock price is too high” would be assigned a neutral sentiment score, even though it contains both positive and negative words.
5. **Return the overall sentiment score:** This score represents the overall sentiment of the tweets, which can be used to predict the stock price movement.
Algorithm End
Example:
Consider the following set of tweets related to AAPL:
* "$AAPL - wall st. kicks off new year on lower note -> URL stock stocks stockaction" (Sentiment score: -0.4) * “rt AT_USER our top 3 trade ideas for 2014 and two $ 1,000 futures wins already in the bank $ gld $ uso $ aapl - - URL” (Sentiment score: 0.2)
* ”$AAPL blackberry and singer alicia keys to part ways URL”(Sentiment score: -0.2)
* “rt AT_USER apple slipped, urban outfitters up on analyst call URL $AAPL” (Sentiment score: -0.3) * “$AAPL - pre-market: apple downgraded on margin concerns ; stocks to start 2014 ... -> URL stock stocks stockaction” (Sentiment score: -0.4)
Using the above algorithm, the overall sentiment score for these tweets would be:
“‘ Overall sentiment = (0.2 * 1 + (-0.4) * 0.8 + (-0.2) * 0.7 + (-0.3) * 0.6 + (-0.4) * 0.5) / (1 + 0.8 + 0.7 + 0.6 + 0.5) = -0.3 ”’
This negative sentiment score indicates that the overall sentiment towards AAPL in these tweets is negative.

Table 23: ReQuest Algorithm: Stock Prediction

You are a highly skilled financial analyst with expertise in predicting stock price movements based on sentiment analysis. Your task is to determine whether the stock price of a given ticker will rise or fall based on the sentiment expressed in tweets related to that ticker.
Instructions:
1. Analyze the provided tweets to assess the overall sentiment.
2. If the overall sentiment is positive, predict that the stock price will rise (output 1).
3. If the overall sentiment is negative, predict that the stock price will fall (output 0).
Note 1: If you find any negative sentiment in the tweets, focus more on that.
Most of the tweets are positive or neutral, so it is crucial to pay close attention to negative sentences. Negative sentiment carries more weight than positive sentiment in this analysis, so prioritize identifying and considering negative expressions but dont forget about positive ones.
Note 2: To ensure consistency and reproducibility, adhere strictly to the following algorithm.Note that the provided example is intended only to demonstrate how the algorithm works; do not base your prediction solely on this example. {algo_prompt}

Table 24: Robustness Check Prompt: Stock Prediction. The algorithm is in Table 23.

f“You are a skilled medical professional.

You are an expert at predicting the level of depression severity of a given text.

Your task is to choose a class from the list of depression severity levels of the text given : {text}.

You can choose only one option from these classes : [{classes}],

You must chose only one level from the list of classes given.

Return only one class of depression severity level for the text given.”

Table 25: Task Prompt: Reddit Suicide Watch (Gemini)

1. Collect a dataset of texts with known severity labels. This dataset can be obtained from a variety of sources, such as online forums, medical records, or research studies.
2. Preprocess the text data. This step involves removing punctuation, stop words, and other irrelevant characters from the text. The data is then converted to lowercase and tokenized.
3. Vectorize the text data. The tokenized text data is then vectorized using a technique such as bag-of-words or TF-IDF. This step converts the text data into a numerical format that can be processed by a machine learning algorithm.
4. Split the dataset into training and testing sets. The dataset is then split into two sets: a training set and a testing set. The training set is used to train the machine learning algorithm, and the testing set is used to evaluate the performance of the algorithm.
5. Train a machine learning algorithm. A machine learning algorithm, such as a support vector machine or a random forest, is then trained on the training set. The algorithm learns to identify the patterns in the text data that are associated with different severity labels.
6. Evaluate the performance of the machine learning algorithm. The performance of the machine learning algorithm is then evaluated on the testing set. The algorithm’s performance is measured using a variety of metrics, such as accuracy, precision, and recall.
7. Deploy the machine learning algorithm. Once the machine learning algorithm has been trained and evaluated, it can be deployed to a production environment. The algorithm can be used to predict the severity of new texts as they are received.

Here is an example of a simple algorithm that can be used to predict the severity of a text:

```
def predict_severity(text):
"""Predicts the severity of a text.
Args:
text: The text to predict the severity of.
Returns:
The predicted severity of the text.
"""
# Preprocess the text.
text = preprocess_text(text)
# Vectorize the text.
vector = vectorize_text(text)
# Load the machine learning model.
model = load_model("severity_prediction_model.pkl")
# Predict the severity of the text.
severity = model.predict(vector)
# Return the predicted severity.
return severity
```

Table 26: ReQuest Algorithm: Reddit Suicide Watch (Gemini)

```
f"You are an expert at following algorithms. You strictly adhere to the algorithm provided.
Your task is to predict the class of depression severity from the text given using this algorithm step by step.
Algorithm: {Reproducible_algorithm}.
You have to predict a class from this list = {classes}.
You must choose a class of severity for this text : text.
You must choose only one class from the list of classes given.
Only return the class of depression severity of the text from the list of classes.
No need to provide any explanation."
```

Table 27: Robustness Check Prompt (Gemini): Reddit Suicide Watch. The algorithm is in Table 26.

```
f"You are a skilled medical professional.
You are an expert at predicting the class of a given text.
Your task is to choose a class from the list of nature of types of the text given : {text}.
You can choose only one option from these classes : [{classes}],
You must chose only one type from the list of classes given.
Return only one class type for the text given."
```

Table 28: Task Prompt: Depression Severity Detection (Gemini)

Step 1: Data Preprocessing

1. Tokenization: The input text is tokenized to break it into individual words or tokens.
2. Stop word removal: Common words that do not contribute to meaning, such as "the," "and," and "is," are removed.
3. Stemming or Lemmatization: Words are reduced to their root forms to capture their core meaning and avoid overfitting on different word forms.

Step 2: Feature Extraction

1. Bag-of-Words (BoW) or TF-IDF: The frequency of each token in the text is calculated to create a feature vector that represents the text. TF-IDF (Term Frequency-Inverse Document Frequency) weighting can be used to emphasize the importance of tokens that are frequent in the given text but rare across a corpus of texts.
2. N-grams: Sequences of adjacent tokens (e.g., bigrams or trigrams) can be extracted to capture phrases or expressions that convey specific sentiments or topics.
3. Sentiment Analysis: The overall sentiment of the text can be analyzed using tools like VADER (Valence Aware Dictionary and Sentiment Reasoner) or TextBlob, which assign valence scores to tokens to determine the emotional tone of the text.

Step 3: Model Training

1. Classification Algorithm: A supervised machine learning algorithm is chosen, such as Logistic Regression, Support Vector Machines (SVM), or Naive Bayes.
2. Training Data: The algorithm is trained on a labeled dataset of text and class labels (e.g., SuicideWatch, Depression, Anxiety).
3. Model Parameters: The algorithm’s hyperparameters (e.g., regularization strength, kernel function) are tuned to optimize performance on a validation set.

Step 4: Model Evaluation

1. Evaluation Metrics: Metrics such as accuracy, precision, recall, and F1-score are used to assess the performance of the model on a test set. Model Selection: The best performing model is selected based on the evaluation results.

Step 5: Classification of New Text

1. Preprocessing: New text is processed using the same preprocessing steps as the training data.
2. Feature Extraction: Features are extracted from the new text using the same techniques used for training.
3. Prediction: The trained model is used to predict the class label of the new text.

Example Algorithm in Python using Scikit-Learn

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
# Import the training data
train_data = pd.read_csv(’training_data.csv’)
# Preprocess the text
train_data[’text’] = train_data[’text’].str.lower()
# Create a bag-of-words feature extractor
vectorizer = CountVectorizer(stop_words=’english’)
# Extract features from the training data
X_train = vectorizer.fit_transform(train_data[’text’])
# Create a Logistic Regression classifier
classifier = LogisticRegression()
# Train the classifier
classifier.fit(X_train, train_data[’class_label’])
# Evaluate the classifier
scores = cross_val_score(classifier, X_train, train_data[’class_label’], cv=5)
print(’Cross-validation score:’, np.mean(scores))
# Predict the class label of a new text
new_text = ’I feel hopeless and like I want to give up on life.’
new_features = vectorizer.transform([new_text])
prediction = classifier.predict(new_features)
print(’Predicted class:’, prediction)
```

Table 29: ReQuest Algorithm: Depression Severity Detection (Gemini)

prompt = f"You are an expert at following algorithms. You strictly adhere to the algorithm provided.
Your task is to predict the class of nature of the given text using this algorithm step by step.
Algorithm: {Reproducible_algorithm}.
You have to predict a class from this list = {classes}.
You must choose a class type for this text : {text}.
You must choose only one class from the list of classes given.
Only return the class of nature of given text from the list of classes.
No need to provide any explanation.

Table 30: Robustness Check Prompt (Gemini): Depression Severity Detection. The algorithm is in Table 29.

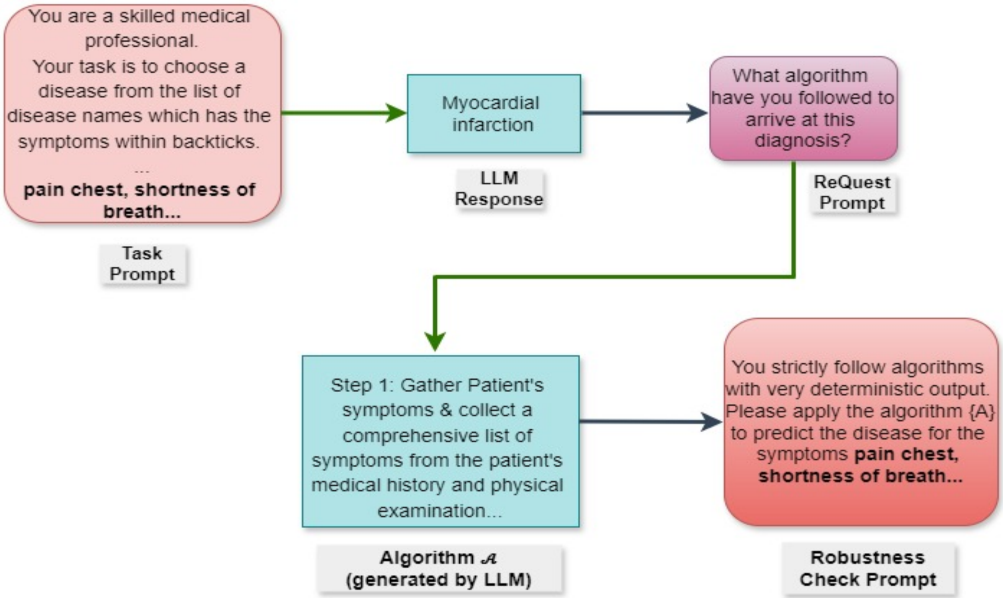
You are a skilled medical professional.
Your task is to choose a disease from the list of disease names which has the symptoms within backticks.
...
pain chest, shortness of breath...
Task Prompt
Myocardial infarction
LLM Response
What algorithm have you followed to arrive at this diagnosis?
ReQuest Prompt
Step 1: Gather Patient's symptoms & collect a comprehensive list of symptoms from the patient's medical history and physical examination...
Algorithm $\mathcal{A}$ (generated by LLM)
You strictly follow algorithms with very deterministic output. Please apply the algorithm {A} to predict the disease for the symptoms pain chest, shortness of breath...
Robustness Check Prompt

Figure 5: An example of the novel prompt regime for the health domain.

Dataset	Task	Example Input (excerpt)	Example Output
[42]	Statute Prediction	..lodged an FIR alleging that the husband and the mother-in-law of the deceased, after the marriage, had been constantly asking for dowry of Rs. 2 lakh from the father of the deceased..	Penalty for demanding dowry
[5]	Human Rights Violation Prediction	...understood that police officers had handcuffed the applicant and hung him and afterwards hit his head against the wall. According to V., police officers had handcuffed the applicant and hung him, and then either he himself...	Human rights violated

Table 31: Examples from the legal datasets

Health Tasks			
Dataset	**Task**	**Example Input (excerpt)**	**Example Output**
[27]	Suicidal tendencies Classification	..hated the fact that I have these mean voices in my head telling me that no one truly cares about me. I have hated the fact that I was so insecure about my weight..	SuicideWatch
[16]	Depression Severity Detection	..then hit me with the newspaper and it shocked me that she would do this, she knows I dońt like play hitting, smacking, striking, hitting or violence..	Moderate
Finance Tasks			
Dataset	**Task**	**Example Input (excerpt)**	**Example Output**
[48]	Stock Price Prediction	..here's how apple could be making a huge push into healthcare..	Price increase (Apple)

Table 32: Examples from the health and finance datasets

Google (GOOG)									
Intra-LLM					Inter-LLM				
Macro F1			Percentage	Ratio	Macro F1	Percentage	Ratio	Percentage	Ratio
baseline (BERTweet)	**Gemini**	**ReQuest algo**	**PerRR_GP_G**	**PreRR_GP_G**	**Gemini → Llama**	**PerRR_GP_L**	**PreRR_GP_L**	**PerRR_GA_L**	**PreRR_GA_L**
0.55	0.45	0.46	97.82 ↑	0.5464	0.48	93.75	0.6473	95.83 ↑	0.6843
baseline (BERTweet)	**Llama**	**ReQuest algo**	**PerRR_LP_L**	**PreRR_LP_L**	**Llama → Gemini**	**PerRR_LP_G**	**PreRR_LP_G**	**PerRR_LA_G**	**PreRR_LA_G**
0.55	0.37	0.35	94.59 ↓	0.9746	0.52	71.15	0.6052	67.30 ↑	0.5502
Amazon (AMZN)									
Intra-LLM					Inter-LLM				
Macro F1			Percentage	Ratio	Macro F1	Percentage	Ratio	Percentage	Ratio
Baseline (BERTweet)	**Gemini**	**ReQuest algo**	**PerRR_GP_G**	**PreRR_GP_G**	**Gemini → Llama**	**PerRR_GP_L**	**PreRR_GP_L**	**PerRR_GA_L**	**PreRR_GA_L**
0.53	0.47	0.49	95.91 ↑	0.4915	0.39	82.97	0.3389	79.59↓	0.6207
Baseline (BERTweet)	**Llama**	**ReQuest algo**	**PerRR_LP_L**	**PreRR_LP_L**	**Llama → Gemini**	**PerRR_LP_G**	**PreRR_LP_G**	**PerRR_LA_G**	**PreRR_LA_G**
0.53	0.43	0.37	86.04 ↓	0.5021	0.49	87.75	0.4131	75.51↓	0.3512

Table 33: Reproducibility: Finance - Google and Amazon datasets

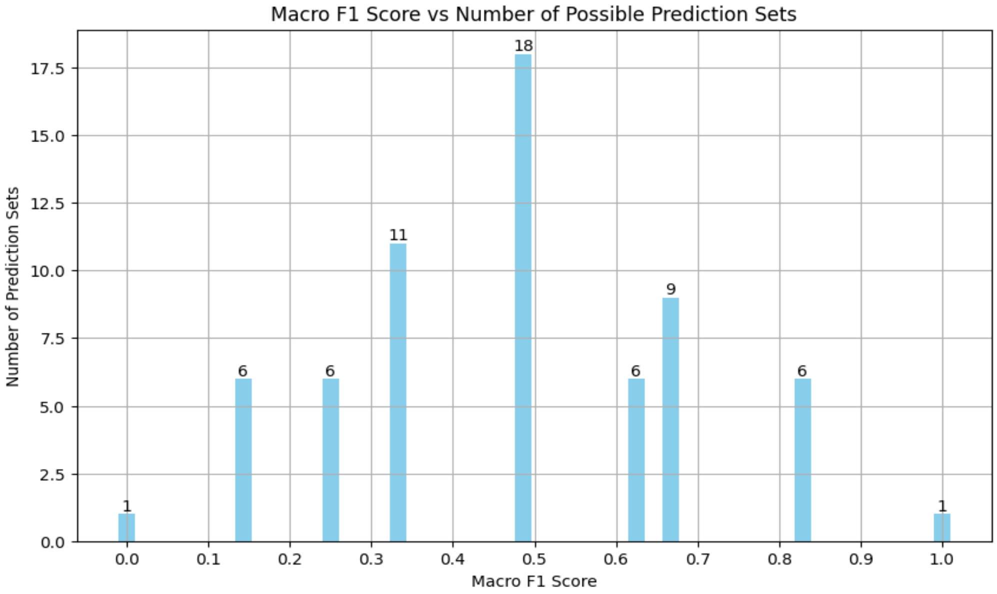

Figure 6: Distribution showing no of possible unique prediction sets vs Macro-F1 for a balanced binary dataset of size 6. The highest ambiguity in Macro-F1 score is towards the centre.

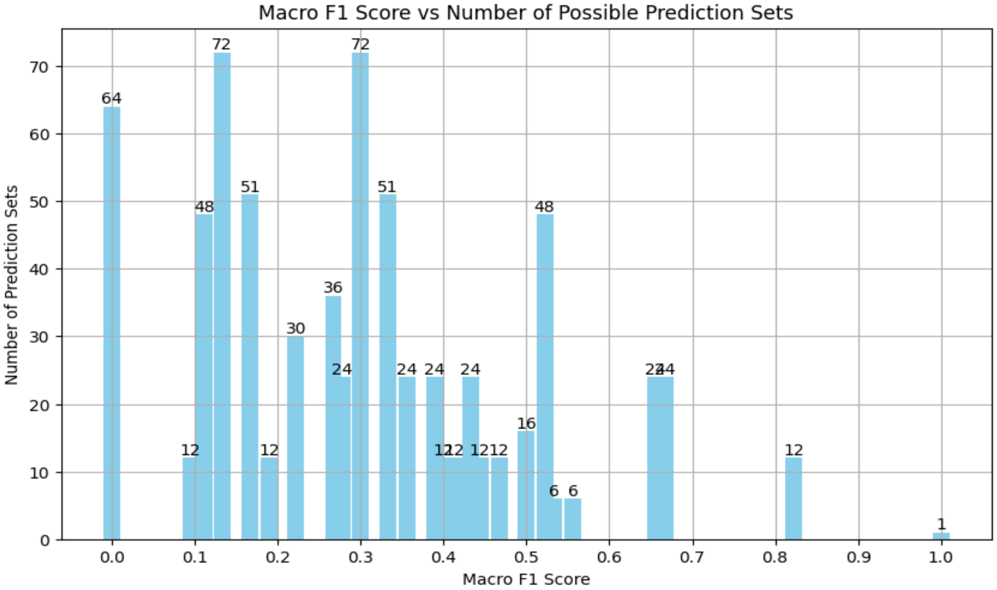

Figure 7: Distribution showing no of possible unique prediction sets vs Macro-F1 for a balanced 3-class multi-class dataset of size 6. Most of the ambiguity in Macro-F1 score is now shifted left as compared to a relatively simpler binary classification problem.