# MZ02 Milestones Compliance Report III: Evaluation Metrics and Testing Methodology Document

## Part 2: Risk-Scoring Algorithm and Evaluation Methodology (Student D)

### 3. Risk-Scoring Algorithm Implementation

This section formalizes the high-level risk-scoring workflow from MCR 2 into a detailed, implementable algorithm.1 It addresses the critical steps of assigning clinically validated severity weights and ensuring the reliability of the AI model's raw probability outputs through calibration.

#### 3.1 Algorithm Architecture and Logic

The core of the risk-scoring system is an algorithm that translates the multi-class probability outputs from the NLP model into a single, interpretable risk score. This score is then mapped to a categorical level that is clinically actionable.

The final risk score, $R$, is formally defined as the weighted sum of the calibrated probabilities for each potential interaction type $i$ identified by the model:

$$R = \sum\_{i=1}^{k} P\_{\text{calibrated}\_i} \times W\_i$$

where $P\_{\text{calibrated}\_i}$ is the calibrated probability of interaction type $i$, and $W\_i$ is the corresponding severity weight for that type.

The following pseudocode outlines the complete logic:

FUNCTION calculate\_risk\_score(model\_logits, temperature, severity\_weights):  
 # Input: Raw logits from the model, learned temperature T, mapping of interaction types to weights.  
 # Output: A categorical risk level ('Low', 'Moderate', 'High', 'Indeterminate').  
  
 # 1. Apply Temperature Scaling to get calibrated probabilities  
 calibrated\_logits = model\_logits / temperature  
 calibrated\_probabilities = softmax(calibrated\_logits)  
  
 # 2. Handle low-confidence predictions  
 max\_prob = max(calibrated\_probabilities)  
 IF max\_prob < 0.2 THEN  
 RETURN 'Indeterminate'  
 END IF  
  
 # 3. Calculate the weighted risk score  
 risk\_score = 0  
 FOR i FROM 1 TO number\_of\_interaction\_types:  
 prob\_i = calibrated\_probabilities[i]  
 weight\_i = severity\_weights[interaction\_type\_i]  
 risk\_score = risk\_score + (prob\_i \* weight\_i)  
 END FOR  
  
 # 4. Categorize the final score  
 IF risk\_score >= 0.7 THEN  
 RETURN 'High'  
 ELSE IF risk\_score >= 0.3 THEN  
 RETURN 'Moderate'  
 ELSE  
 RETURN 'Low'  
 END IF  
END FUNCTION

This algorithm incorporates a crucial error handling step. If the model's highest confidence for any single interaction type is below a threshold of 0.2, the system returns an 'Indeterminate' status.1 This safety measure prevents the system from providing a definitive risk level based on a highly uncertain prediction, instead flagging it for necessary manual review by a clinician.

#### 3.2 Severity Weight Finalization and Justification

The clinical validity of the risk score $R$ is entirely dependent on the principled assignment of the severity weights, $W\_i$. These weights cannot be arbitrary; they must be grounded in established clinical standards.

The system will adopt the three-tiered severity classification system used by DrugBank, a comprehensive and widely respected drug information database: minor, moderate, and major.17 The definitions are as follows:

* **Minor:** The interaction may be clinically significant, but the benefits of use generally outweigh the risks. Observation and potential dose adjustments may be needed.
* **Moderate:** The benefits of continued use should be evaluated. Actions like observation, dose changes, or medication changes should be considered.
* **Major:** The combination may cause more harm than benefit. Alternative medications should be strongly considered.

A critical step in this process is to create a formal mapping between the interaction *types* predicted by the AI model and these clinical severity levels. The model, trained on the DDIExtraction 2013 corpus, will learn to classify interactions into types such as "pharmacokinetic," "pharmacodynamic," or "adverse effect".1 This mapping provides the essential link between the model's technical output and the algorithm's clinical meaning. This is achieved by analyzing the annotation guidelines of the training data and cross-referencing them with examples and definitions from DrugBank.

The numerical weights are assigned non-linearly to reflect the escalating clinical danger associated with each severity level. A major interaction is not merely three times as risky as a minor one; its potential for harm is substantially greater.

| **Model Output Interaction Type** | **Corresponding DrugBank Severity** | **Assigned Numerical Weight (Wi​)** | **Justification/Example** |
| --- | --- | --- | --- |
| Advice/Recommendation | Minor | 0.2 | Interaction requires observation (e.g., "monitor blood pressure"). |
| Mechanism (Pharmacodynamic) | Moderate | 0.5 | Interaction affects the drug's mechanism of action, potentially requiring dose adjustment. |
| Effect (Adverse/Synergistic) | Moderate | 0.6 | Interaction leads to an increased risk of a known adverse effect. |
| Mechanism (Pharmacokinetic) | Major | 0.9 | Interaction significantly alters a drug's metabolism (e.g., CYP3A4 inhibition), leading to potential toxicity. |
| Intravenous/Contraindicated | Major | 1.0 | The interaction is explicitly noted as high-risk or contraindicated. |

#### 3.3 Probability Calibration with Temperature Scaling

Modern neural network architectures, including the PubMedBERT model used in this project, are known to be poorly calibrated. They often produce overconfident softmax probability scores that do not represent the true likelihood of an event.19 For example, a predicted probability of 0.95 from an uncalibrated model does not mean there is a 95% chance of an interaction occurring. Relying on these raw, uncalibrated probabilities would make the weighted risk score calculation unreliable and clinically unsafe.

To address this, temperature scaling will be implemented as a post-processing step on the model's outputs. This technique works by dividing the logits (the raw, pre-softmax outputs of the model's final layer) by a single, learned scalar parameter, $T$ (the "temperature"), before applying the softmax function.19

$$P\_{\text{calibrated}\_i} = \frac{\exp(z\_i / T)}{\sum\_{j=1}^{k} \exp(z\_j / T)}$$

When $T > 1$, this operation "softens" the probability distribution, reducing the overconfidence of the highest-probability class and increasing the probabilities of the lower-ranked classes. The optimal value for $T$ is found by optimizing it on a held-out validation set to minimize the Negative Log-Likelihood (NLL).20

The following Python code snippet, using the PyTorch framework, demonstrates a prototype implementation. This approach involves creating a wrapper class for the trained model that adds a learnable temperature parameter, which is then tuned on the validation data.

Python

import torch  
from torch import nn, optim  
from torch.nn import functional as F  
  
class ModelWithTemperature(nn.Module):  
 """  
 A thin wrapper for a trained model that adds a learnable temperature parameter.  
 """  
 def \_\_init\_\_(self, model):  
 super(ModelWithTemperature, self).\_\_init\_\_()  
 self.model = model  
 # Initialize temperature to 1.0  
 self.temperature = nn.Parameter(torch.ones(1) \* 1.5)  
  
 def forward(self, input):  
 logits =...[source](https://www.kaggle.com/code/anonamename/temperature-scaling)  
  
 def set\_temperature(self, valid\_loader):  
 """  
 Tune the temperature of the model (using the validation set).  
 We're going to set it to optimize NLL.  
 """  
 self.cuda()  
 nll\_criterion = nn.CrossEntropyLoss().cuda()  
 ece\_criterion = \_ECELoss().cuda()  
  
 ...[source](https://www.kaggle.com/code/anonamename/temperature-scaling)  
  
 # Optimize the temperature parameter w.r.t. NLL  
 optimizer = optim.LBFGS([self.temperature], lr=0.01, max\_iter=50)  
  
 def eval():  
 optimizer.zero\_grad()  
 loss = nll\_criterion(self.temperature\_scale(logits), labels)  
 loss.backward()  
 return loss  
 optimizer.step(eval)  
   
 return self

This implementation aligns with best practices found in established open-source projects and libraries such as net:cal, ensuring a robust and effective calibration process.21

### 4. Formal Evaluation Metrics and Testing Methodology

This section establishes a comprehensive framework for evaluating the AI model's performance. The methodology is designed to be statistically sound, clinically relevant, and capable of providing deep insights into the model's operational characteristics, moving beyond simplistic metrics to a more nuanced assessment of its strengths and weaknesses.

#### 4.1 Performance Metrics for Imbalanced Classification

A fundamental challenge in DDI detection is severe class imbalance; in any given text corpus or clinical setting, the number of non-interacting drug pairs vastly exceeds the number of interacting pairs. In such a scenario, accuracy is a dangerously misleading metric. A naive model that always predicts "no interaction" could easily achieve over 99% accuracy while being completely useless for its intended clinical purpose, as it would fail to detect any actual risks.24 Therefore, the evaluation will focus on metrics that are robust to class imbalance and reflect clinically meaningful performance.

* **Precision:** Calculated as $\frac{TP}{TP + FP}$, precision answers the question: "Of all the interactions the model flagged, what percentage were actually correct?" High precision is critical for gaining clinician trust and preventing "alert fatigue," where users begin to ignore system warnings due to a high rate of false alarms.
* **Recall (Sensitivity):** Calculated as $\frac{TP}{TP + FN}$, recall answers the question: "Of all the true interactions that existed in the data, what percentage did the model successfully identify?" High recall is paramount for patient safety, as failing to detect a genuine interaction (a false negative) could have severe clinical consequences.
* **F1-Score:** Calculated as $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$, the F1-score is the harmonic mean of precision and recall. It provides a single, balanced measure of a model's performance, which is particularly useful when both precision and recall are of equal importance.
* **Precision-Recall Area Under Curve (PR-AUC):** This will be the primary metric used for model selection and hyperparameter optimization.1 The PR curve plots precision against recall at various decision thresholds. Unlike the ROC curve, whose baseline is fixed at 0.5, the baseline for a PR curve is determined by the class ratio (i.e., the fraction of positives). This makes PR-AUC a much more informative and sensitive metric for imbalanced datasets, as improvements over the random baseline are more difficult to achieve and thus more meaningful.27

#### 4.2 Cross-Validation Strategy

To ensure a robust and unbiased estimate of the model's generalization performance, a **stratified k-fold cross-validation** strategy will be employed, with $k=10$.29 In this procedure, the dataset is divided into 10 folds. The model is then trained 10 times, each time using 9 of the folds for training and the remaining 1 for validation. The final performance metrics are averaged across all 10 runs.

The use of **stratification** is critical. A standard k-fold split on an imbalanced dataset could randomly result in some validation folds containing very few, or even zero, positive-class instances. This would lead to unstable and unreliable performance estimates.30 Stratified sampling explicitly preserves the percentage of samples for each class within each fold, ensuring that every training and validation split is representative of the overall data distribution. This leads to a more stable, reliable, and trustworthy evaluation of the model's true performance.

#### 4.3 Error Analysis Protocol

Quantitative metrics provide a high-level summary of performance but do not explain *why* a model fails. A systematic error analysis protocol is essential for identifying patterns in model mistakes, which in turn provides actionable directions for targeted model improvement.

The methodology will involve a manual review of a statistically significant sample of false positive (FP) and false negative (FN) predictions from the held-out test set. Each error will be categorized according to a structured taxonomy adapted from established clinical NLP research.32

* **Proposed Error Taxonomy for DDI Extraction:**
  + **Linguistic Errors:**
    - *Negation/Uncertainty Misinterpretation:* The model fails to correctly interpret phrases like "no significant interaction," "unlikely to interact," or "risk may be elevated."
    - *Complex Sentence Structure:* The model fails to extract a relationship from a sentence that is syntactically complex, containing multiple subordinate clauses or distant drug mentions.
  + **Contextual/Semantic Errors:**
    - *Implicit Interaction:* An interaction is implied by the context (e.g., "requires therapeutic drug monitoring") but is not stated with explicit trigger words like "interacts" or "inhibits."
    - *Conditional Interaction:* The model misses a condition required for the interaction (e.g., "in patients with CYP2C19 poor metabolizer status").
  + **Upstream Component Errors:**
    - *Entity Recognition Failure:* The error stems not from the relation extraction component but from an upstream failure to correctly identify or bound the drug names in the text.
  + **Data Annotation Errors:**
    - *Ambiguous or Incorrect Label:* The "error" is discovered to be a result of an incorrect or ambiguous ground-truth label in the DDIExtraction 2013 dataset itself.

This structured analysis transforms evaluation from a passive measurement into an active feedback loop. For example, if the analysis reveals that a high percentage of false negatives are due to the model's failure to understand negated statements, this provides a clear directive for the next development cycle: augment the training data with more diverse examples of negated interactions. Similarly, identifying frequent failures on implicit interactions might suggest the need for a more sophisticated model architecture or feature engineering. This process of identifying, categorizing, and learning from errors is fundamental to the iterative refinement and improvement of the clinical NLP model.

#### 4.4 Clinical Relevance and Safety Testing Plan

Offline performance metrics, while essential, do not fully capture a model's utility and safety in a real-world clinical setting. Therefore, a forward-looking plan is proposed to assess the model's clinical relevance and to de-risk its eventual implementation. This plan is inspired by modern evaluation frameworks for clinical AI that emphasize testing in realistic, context-rich scenarios.33

* Phase 1: Heuristic Evaluation with a Clinical Domain Expert:  
  In the short term, a qualitative review of the model's outputs will be conducted with a collaborating pharmacist or physician. A curated set of 50-100 predictions—including true positives, false positives, and false negatives—will be presented to the expert. The expert will provide structured feedback on several axes:
  + **Clinical Correctness:** Does the model's finding align with their expert knowledge and current clinical guidelines?
  + **Actionability:** Is the information provided (risk level, interaction type, evidence) sufficient and clear enough to inform a clinical decision?
  + Trustworthiness and Interpretability: Is the supporting evidence highlighted by the model relevant and convincing? Does the model's reasoning appear sound?  
    This phase will provide crucial early feedback on whether the model's outputs are not just technically correct but also clinically useful.
* Phase 2: Simulated Use Case Study (Future Work Proposal):  
  For a more rigorous assessment, a future study will be designed where clinicians interact with the deployed system in a simulated environment. This would involve creating a set of realistic, challenging patient case vignettes. Participating clinicians would use the tool to check for DDIs as part of their simulated workflow. This methodology is inspired by established clinical evaluation benchmarks like the Objective Structured Clinical Examination (OSCE) and simulation platforms like SDBench.33 Key metrics to be collected would include:
  + **Decision Congruence:** How often does the clinician's final assessment of DDI risk align with the system's recommendation?
  + **Time to Decision:** Does the tool demonstrably reduce the time required to research and assess potential DDIs compared to manual methods?
  + **User Feedback:** Qualitative data would be collected through post-simulation surveys and interviews to gauge user satisfaction, perceived impact on cognitive load, and confidence in the system's recommendations.

This two-phase plan acknowledges the critical translational gap between developing a performant algorithm and deploying a safe, effective, and trusted clinical tool, demonstrating a mature understanding of the lifecycle of clinical AI systems.36

#### Works cited

1. lowlevelarcdesignreport3.drawio (2).pdf
2. Google Kubernetes Engine pricing | Google Cloud, accessed October 30, 2025, <https://cloud.google.com/kubernetes-engine/pricing>
3. Google Kubernetes Engine (GKE) – The Definitive Pricing Guide - Economize Cloud, accessed October 30, 2025, <https://www.economize.cloud/blog/google-kubernetes-engine-pricing-guide/>
4. GCP Kubernetes Pricing - Cost Breakdown & Savings Tips - Pump, accessed October 30, 2025, <https://www.pump.co/blog/gcp-kubernetes-pricing>
5. Vertex AI pricing | Google Cloud, accessed October 30, 2025, <https://cloud.google.com/vertex-ai/pricing>
6. Google Cloud SQL: Pricing and Cost Optimization - ProsperOps, accessed October 30, 2025, <https://www.prosperops.com/blog/google-cloud-sql-pricing-and-cost-optimization/>
7. Cloud SQL pricing | Google Cloud, accessed October 30, 2025, <https://cloud.google.com/sql/pricing>
8. Cloud Storage pricing - Google Cloud, accessed October 30, 2025, <https://cloud.google.com/storage/pricing>
9. Pricing examples | Google Cloud Documentation, accessed October 30, 2025, <https://cloud.google.com/storage/pricing-examples>
10. Free cloud features and trial offer | Google Cloud Free Program, accessed October 30, 2025, <https://docs.cloud.google.com/free/docs/free-cloud-features>
11. What are the key differences between NVIDIA A100 and T4 GPUs in ..., accessed October 30, 2025, [https://massedcompute.com/faq-answers/?question=What%20are%20the%20key%20differences%20between%20NVIDIA%20A100%20and%20T4%20GPUs%20in%20terms%20of%20performance%20and%20power%20efficiency?](https://massedcompute.com/faq-answers/?question=What+are+the+key+differences+between+NVIDIA+A100+and+T4+GPUs+in+terms+of+performance+and+power+efficiency?)
12. NVIDIA T4 vs. NVIDIA A100 Comparison: Which GPU Should You Choose for AI and Data Center Workloads? - server-parts.eu, accessed October 30, 2025, <https://www.server-parts.eu/post/nvidia-t4-vs-a100-gpu-comparison-ai-deep-learning-data-centers>
13. How does the training time of BERT models compare on T4 and A100 GPUs?, accessed October 30, 2025, [https://massedcompute.com/faq-answers/?question=How%20does%20the%20training%20time%20of%20BERT%20models%20compare%20on%20T4%20and%20A100%20GPUs?](https://massedcompute.com/faq-answers/?question=How+does+the+training+time+of+BERT+models+compare+on+T4+and+A100+GPUs?)
14. DEEP LEARNING PERFORMANCE GUIDE - M Computers, accessed October 30, 2025, <https://mcomputers.cz/wp-content/uploads/2020/12/3_NVIDIA_Ralph_Hinsche.pdf>
15. Pricing | Spot VMs | Google Cloud, accessed October 30, 2025, <https://cloud.google.com/spot-vms/pricing>
16. Use Spot VMs with training | Vertex AI - Google Cloud Documentation, accessed October 30, 2025, <https://docs.cloud.google.com/vertex-ai/docs/training/use-spot-vms>
17. Articles | DrugBank Help Center, accessed October 30, 2025, <https://dev.drugbank.com/guides/drug_interactions/parameters>
18. Drug Interaction Checker | DrugBank Online, accessed October 30, 2025, <https://go.drugbank.com/drug-interaction-checker>
19. Temperature scaling - AWS Prescriptive Guidance, accessed October 30, 2025, <https://docs.aws.amazon.com/prescriptive-guidance/latest/ml-quantifying-uncertainty/temp-scaling.html>
20. Calibrating Neural Networks - Geoff Pleiss, accessed October 30, 2025, <https://geoffpleiss.com/blog/nn_calibration.html>
21. gpleiss/temperature\_scaling: A simple way to calibrate your neural network. - GitHub, accessed October 30, 2025, <https://github.com/gpleiss/temperature_scaling>
22. EFS-OpenSource/calibration-framework: The net:cal ... - GitHub, accessed October 30, 2025, <https://github.com/EFS-OpenSource/calibration-framework>
23. saurabhgarg1996/calibration: Simple implementation of Temperature and Vector Scaling in python - GitHub, accessed October 30, 2025, <https://github.com/saurabhgarg1996/calibration>
24. Classification: Accuracy, recall, precision, and related metrics ..., accessed October 30, 2025, <https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall>
25. How to Handle Imbalanced Datasets in Predictive Modeling for Accurate Results?, accessed October 30, 2025, <https://learn.microsoft.com/en-us/answers/questions/2243657/how-to-handle-imbalanced-datasets-in-predictive-mo>
26. Imbalanced class distribution and performance evaluation metrics: A ..., accessed October 30, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC10688675/>
27. ROC AUC vs Precision-Recall for Imbalanced Data - MachineLearningMastery.com, accessed October 30, 2025, <https://machinelearningmastery.com/roc-auc-vs-precision-recall-for-imbalanced-data/>
28. ROC and precision-recall with imbalanced datasets, accessed October 30, 2025, <https://classeval.wordpress.com/simulation-analysis/roc-and-precision-recall-with-imbalanced-datasets/>
29. codesignal.com, accessed October 30, 2025, <https://codesignal.com/learn/courses/building-and-evaluating-text-classifiers-in-python/lessons/mastering-cross-validation-for-text-classification-in-python#:~:text=The%20most%20common%20type%20of,subset%20as%20the%20validation%20set.&text=In%20Python%2C%20we%20can%20easily,specifically%20the%20cross_val_score()%20function.>
30. Understand K-Fold Cross-Validation:A Step-by-Step Guide ..., accessed October 30, 2025, <https://www.digitalocean.com/community/tutorials/k-fold-cross-validation-python>
31. 3.1. Cross-validation: evaluating estimator performance - Scikit-learn, accessed October 30, 2025, <https://scikit-learn.org/stable/modules/cross_validation.html>
32. A taxonomy for advancing systematic error analysis in multi-site ..., accessed October 30, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC11187420/>
33. New Benchmarks Envision the Future of AI in Healthcare | Scale, accessed October 30, 2025, <https://scale.com/blog/healthcare-benchmarks>
34. Evaluating AI in context: Which LLM is best for real health care needs? - Stanford Medicine, accessed October 30, 2025, <https://med.stanford.edu/news/insights/2025/04/ai-artificial-intelligence-evaluation-algorithm.html>
35. How AI-based simulation testing is revolutionising healthcare - Pearson VUE, accessed October 30, 2025, <https://www.pearsonvue.com/us/en/about/news/highlights/how-ai-based-simulation-testing-is-revolutionising-healthcare.html>
36. The Integration of Artificial Intelligence into Clinical Practice - MDPI, accessed October 30, 2025, <https://www.mdpi.com/2813-0464/3/1/2>
37. Monitoring performance of clinical artificial intelligence in health care: a scoping review, accessed October 30, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC11630661/>