ChatGPT

Plan for Final Report Revision: Hierarchical Classification of Academic Computing Papers

To produce a coherent and comprehensive final report, we will restructure and expand each chapter according to the six required sections. The plan below outlines the content and focus for each chapter, how to integrate the chronological model evolution (v2.3 through v9.0), and ways to address the feedback about methodology simplicity, model complexity, and category expansion. We also note where to adjust word counts (expand or trim) to meet the guidelines (Introduction ≤1000 words, Literature ≤2500, Design ≤2000, Implementation ≤2000, Evaluation ≤2500, Conclusion ≤1000, total ≤9500).

Chapter 1: Introduction (≤1000 words)

- **Problem Statement & Motivation**: Begin with a clear, concise statement of the problem the system solves (as suggested in feedback) [1]. Emphasize the challenge of navigating the explosive growth of computing literature across disciplines and methodologies, and the need for an automated hierarchical classifier [2] [3]. For example: *"Researchers struggle to find relevant work across blurred boundaries of CS, IS, IT – this project fills that gap by classifying papers by discipline, subfield, and research methodology."*

- **Scope of Solution**: Briefly describe the three-tier classification structure (Discipline → Subfield → Methodology) with a visual or description. You can reuse the succinct ASCII diagram from the draft [4] to illustrate the hierarchy, ensuring a layperson can grasp it. Mention the system classifies into 3 disciplines (CS/IS/IT), 15 subfields, and 3 research methodologies (Qualitative, Quantitative, Mixed) [5].

- **Project Objectives**: List the key goals of the project, such as accuracy targets and methodological soundness. For instance, state that the aim was to achieve at least ~85% accuracy in each classification layer and to establish a robust pipeline with proper validation [6]. Also highlight the educational goal of deriving ML best practices from this project's journey [7].

- **Evolution Overview**: Introduce the evolution of the classification models through major versions, to set context for later chapters. Summarize how the approach evolved from initial baseline models to the final production system. A small table or narrative can highlight Version **v2.4** (early baseline with transformer embeddings), **v6.0** (high accuracy but flawed), and **v7.0** (slightly lower accuracy but methodologically correct) [8] [9]. For example:

Version	Key Approach	Accuracy (sample)	Status
v2.4 (Baseline)	SPECTER embedding + XGBoost	~75% (Methodology) [10]	*Superseded* (simple model)
v6.0 (Complex)	Multi-model ensemble, 11.5k features	94.77% (Discipline) [11], 91.87% (Methodology)	*Flawed* (data leakage)

Version	Key Approach	Accuracy (sample)	Status
v7.0 (Final)	Single XGBoost, 3k features, proper validation	92.32% (Discipline) [11], 86.92% (Methodology)	**Production** (sound)

This table (as in the draft intro) concisely shows the progression and will help the reader see that while v6.0 had higher raw accuracy, v7.0 is the reliable solution [12] [13]. We will update it if needed (for instance, ensure the accuracy values and status are up-to-date).

- **Novelty and Insights Teaser**: End the introduction by foreshadowing the critical insight gained: that methodological rigor was more important than raw model complexity or metric-chasing. Note that the project uncovered a major **data leakage issue** in early high-performing models and solved it, providing a cautionary tale for ML research [14] [15]. Also mention in one sentence that we even experimented with expanding methodology categories from 3 to 8 in a later version, learning that more complexity isn't always better – this addresses the "3 paradigms too simple" concern and piques interest (details will follow in Conclusion) [16] [17]. Keeping this teaser brief will avoid cluttering the introduction with too much detail.

(The introduction in the draft is already strong, but we will trim any overly long background and ensure the first paragraph clearly states the problem and solution approach [1]. We'll incorporate a problem statement at the very start, add a short pipeline diagram if possible (as suggested in feedback), and include the updated table of versions.)

Chapter 2: Literature Review (≤2500 words)

- **2.1 Background and Significance**: Start by situating the project in the context of academic text classification. Reiterate why classifying research papers is important (information overload, interdisciplinary nature) with references to existing classification systems' shortcomings (e.g., ACM CCS or IEEE taxonomy inadequacies) [18]. This sets up the need for our approach.

- **2.2 Text Classification Techniques**: Summarize general text classification literature across traditional ML and deep learning. Use the structure from the draft: discuss the evolution from statistical methods to ML to deep learning [19] [20]. Emphasize that while transformers (BERT, etc.) are state-of-the-art, classic methods like TF-IDF with XGBoost can still excel on domain-specific tasks [21] [22]. Cite Zhang et al. or similar literature that support the competitiveness of classical methods for certain tasks [21], to justify our approach. We keep this section concise enough and ensure it flows into our project's choices (for instance, noting that our final model ended up using TF-IDF + XGBoost, aligning with literature that simpler models can work well on moderate-sized data [21] [23]).

- **2.3 Academic Text Classification Challenges**: Discuss what makes classifying academic papers unique. Leverage literature on academic text characteristics: specialized vocabulary, class imbalance, fuzzy boundaries between fields [24] [25]. Connect this to our task (e.g., distinguishing CS vs IS vs IT, and subfields, is tricky because terms overlap but contexts differ [26] [27]). If available, mention any prior work on classifying research papers by discipline or topic (the ACM Digital Library or arXiv categorization can be examples, though not exact parallels to our hierarchical approach). Also, discuss *methodology classification* in literature: it's relatively uncommon to classify papers by research method, so highlight the novelty of that dimension in our work (we may note that existing catalogs focus on topics, not whether a study is qualitative or quantitative, making our work distinct).

- **2.4 Hierarchical and Multi-Label Classification**: Include a subsection on hierarchical classification techniques, since our system is inherently hierarchical (discipline → subfield). Briefly review strategies like top-down classification and error propagation issues. If there's literature (e.g., on hierarchical text classification algorithms), cite it. This sets context for why we chose a sequential pipeline (we first classify discipline, then choose the appropriate subfield classifier) and how we mitigate errors cascades (like confidence-based routing mentioned later). We can reference our system's architecture here conceptually to illustrate this approach is aligned with known methods (for example, mention how subfield classification is *conditioned* on discipline classification output, which is a form of hierarchical classification consistent with literature best practices).

- **2.5 Related Work in Research Paper Classification**: Survey any known systems or studies (if any) that tried to classify academic publications by field or methodology. Perhaps mention the ACM CCS taxonomy usage in digital libraries or machine learning approaches for subject classification in bibliometrics. If there's no directly comparable work on methodology classification, state that this project breaks new ground in that aspect, highlighting originality.

- **2.6 Validation and ML Methodology in Literature**: Given our project's focus on proper validation, mention literature on common pitfalls like data leakage. We can cite sources or well-known anecdotes (e.g., papers or blog posts on leakage issues) to frame that our experience is a known concern in ML. This reinforces that our rigorous approach in v7.0 aligns with expert recommendations. For example, point out that our finding echoes known best practices: always use a proper holdout and cross-validation to trust results (we may not have external citation here, but we can reference that our logs/doc emphasize it [28] [29]).

(The draft literature review already covers fundamentals thoroughly [19] [21]. We'll integrate any missing pieces such as hierarchical classification and any references to prior attempts to classify academic content. We should also weave in how our project's findings relate to existing knowledge – e.g., "While many works focus on improving model architectures, our experience underscores the lesser-discussed aspect of validation methodology." This sets the stage for the unique insight of our project.)

Chapter 3: Design (≤2000 words)

This chapter will describe how the system was designed and how the design evolved over time. We will organize it into clear subsections:

- **3.1 Design Overview and Evolution**: Provide an introduction to the system's high-level design and note that it went through multiple iterations to reach the final form [30]. We should explicitly mention the key versions (v2.x, v6.0, v7.0, v8.0, v9.0) and what design philosophy each represented:
- *Baseline (v2.x)* – simple models with transformer embeddings (SPECTER) + XGBoost, separate classifiers per task [31].
- *Complex pipeline (v6.0)* – an ensemble of models, heavy feature engineering, aiming for maximum accuracy [32].
- *Refined pipeline (v7.0)* – simplified, unified approach focusing on methodological correctness and generalization [33].

- Mention that we also attempted *experimental designs* beyond v7: a *two-stage methodology classifier* in v8.0 and an *expanded category classifier* in v9.0, although these were ultimately not adopted in production. (We will detail their design differences later in this chapter or in Evaluation/Conclusion as appropriate.)

- **3.2 System Architecture**: Describe the overall architecture of the final system (v7.0). This includes the three-tier classification flow: first Discipline, then Subfield, then Methodology. We should include or refer to a diagram of this hierarchy (like **Figure 8** in the draft) [34] [35]. Emphasize that subfield classification is *dependent* on discipline (e.g., a paper first goes to the CS subfield model if classified as CS) [26]. Also note the total classification space is 3 × 15 × 3 = 135 combinations, highlighting complexity [36] [37]. This addresses the "combinatorial complexity" concern by showing the system's design had to handle many categories and dependencies, not a trivial 3-class problem [37].

- **3.3 Feature Representation & Model Choices**: Explain the feature engineering and model selection in design terms. Contrast the approaches in different versions:

- **v2.x**: used pre-trained SPECTER embeddings (768-dim) directly as features into XGBoost [38]. Simpler pipeline (no custom features; one model per task). Easy to implement but we noticed limitations (e.g., moderate accuracy ~75% for methodology) [39].
- **v4.x/v5.x**: experimented with transformer fine-tuning (SciBERT with LoRA) and ensemble with XGBoost [40] [41]. The design became hybrid (neural + ML). Mention focal loss for class imbalance and targeted augmentation as design features in v4/v5 [40] [42]. Outcome: transformer-based component underperformed the simpler XGBoost; by v5.0, a weighted ensemble of SciBERT+XGBoost was designed but yielded only marginal gains [43] [44].
- **v6.0**: a key design shift. Describe the *multi-vectorizer + ensemble* architecture. We had an **OptimizedFeatureExtractor** with multiple TF-IDF configurations (word n-grams, char n-grams, etc.) and also handcrafted domain features (like keyword counts) [45] [46]. These fed into an ensemble of XGBoost models (7 models with different seeds) to boost performance [47] [48]. The design goal was to maximize accuracy (and indeed v6.0 design achieved ~94-95% discipline accuracy) at the cost of complexity. We will cite the logs illustrating this complexity: e.g., *11,500 TF-IDF features + 46 domain features used* [49] [50], and an ensemble approach was considered "production-ready" at that time [51].
- **v7.0**: highlight the design simplifications made. All data flows through a unified pipeline with proper train/val/test splits. Feature extraction was streamlined to 3,000 TF-IDF features total (e.g. using just two vectorizers: 1–2 gram words and 3–5 char ngrams) [45] [52], and we dropped the complex ensemble in favor of a single XGBoost model per classifier [53] [48]. The design choice here was guided by the *methodological corrections* needed – ensure no information from test leaks into training, etc. We can reference the log for v7.0 design where data splitting and feature reduction were core design principles [54] [55].

- We will mention that the v7.0 design still retained the hierarchical structure and *modularity*: separate classifiers for each sub-problem but integrated pipeline. Also note any design decisions about **post-processing**: in the unified pipeline, we implemented confidence-based corrections (to catch obvious errors like if a CS paper is misrouted, etc.) as a design element [56] [57]. This shows the design's maturity in handling edge cases.

- **3.4 Key Design Decisions and Trade-offs**: Discuss lessons learned from these design evolutions:

- Simpler vs Complex: We found that a simpler design (v7.0 single-model, fewer features) nearly matched or exceeded the complex design's real-world performance [58] [59]. This justifies why the final design reverted to a more straightforward architecture.
- Handling Imbalanced Data: In design, we attempted different strategies (class weighting in v2.5, focal loss in v4.0, data augmentation in v4/v5/v6). Note which ones we kept in final: e.g., in v7.0 we did moderate augmentation and ensured stratified splits, instead of aggressive oversampling or exotic losses [60] [61]. We can cite the logs: v5.0 design did WordNet augmentation 30% and

class-balancing to 80% of majority [42] [62], whereas v7.0 design used a 50% augmentation on training only [63].

- **Two-Stage vs One-Stage Methodology Classifier**: Highlight a specific design experiment for methodology classification. Early on, we considered a two-stage design (first detect if a paper is "Mixed" or not, then classify Qual vs Quant) in v2.6 [64]. This design reappeared in v8.0. We should describe this as a potential design improvement that turned out to be based on a false premise. In v8.0's design, we reverted to a two-model approach under the assumption that v7.0's single model missed Mixed cases [65] [66]. The two-stage design had a threshold for what constitutes "Mixed" (as in v2.6, threshold 0.15 on first model confidence) [67]. However, we learned that v7.0's one-stage model was already performing well on Mixed, so this additional complexity was unnecessary. We'll mention this to show we *designed* and tried a more sophisticated methodology classifier (addressing the "only 3 paradigms" concern through design experimentation) but eventually **reverted** to the simpler one-stage design. (The outcome of v8.0 will be detailed in Evaluation, but the design chapter can note the approach.)

- **v9.0 Experimental Design**: Similarly, briefly describe the design of the v9.0 methodology classifier with 8 categories. For design purposes, note that it introduced a new label schema (eight detailed research method categories) and required careful data annotation and augmentation to support it [68]. Technically, its implementation was an extension of v7.0's pipeline (also using OptimizedFeatureExtractor and XGBoost) but applied to a finer-grained label set [69] [70]. We should state that *design-wise*, v9.0 was a "maximally complex label space" scenario to test the limits of our classifier. This shows the committee that we did explore an evolution beyond the simple 3 classes – integrating that into our design discussion demonstrates responsiveness to feedback. However, foreshadow that this design was discontinued due to strategic reasons (to be fully explained later) [71].

- **3.5 Final System Design Summary**: End the design chapter with a summary of the final system's components and why they were chosen:

- Reiterate the final pipeline (v7.0) architecture with perhaps a diagram comparing **v2.4 vs v6.0 vs v7.0 architectures** (the draft's Figure 9) [72] [53]. This diagram already succinctly shows how we went from a SPECTER+XGBoost (v2.4) to a 7-model ensemble with multi-TFIDF (v6.0) to an optimized single XGBoost (v7.0) [73] [74]. We will include it or convert it into a simple flowchart image for clarity (as feedback suggested adding a pipeline figure).
- List the major components of the final design:
    1. A **data preprocessing module** (responsible for cleaning text, splitting data properly, and augmentation on training only),
    2. An **feature extraction module** (OptimizedFeatureExtractor with two TF-IDF vectorizers) [52] [75],
    3. Three **classifier models** (discipline XGBoost, subfield classifiers for each discipline – logistic/XGB, and methodology XGBoost) trained with cross-validation.
    4. An **integration logic** that connects these (e.g., pipeline code that takes an abstract through discipline → subfield → methodology, and applies post-processing like confidence threshold for corrections).
- Emphasize design choices that ensure robustness: using a shared random seed for splits (documented in code) and consistent train/val/test across all classifiers for fairness [60] [76].

(The design chapter in the draft covers much of this evolution and architecture. We will ensure to incorporate v8.0 and v9.0 design notes as described, since the draft focused mostly on v2.4, v6.0, v7.0. By including v8 and v9 in the design story (albeit briefly), we demonstrate we considered and designed more complex classifiers even if they weren't kept. This addresses the feedback about exploring beyond 3 methodologies and shows critical evaluation of complexity vs simplicity. We should keep the design chapter focused on structure *and*

components*, leaving detailed performance results or code for the Implementation and Evaluation chapters. Any redundant explanation already in Implementation can be trimmed here to avoid overlap.)*

Chapter 4: Implementation (≤2000 words)

This chapter will detail how we realized the design in practice. It should describe the development of the system, the code structure, and how each major component was implemented. We will also narrate the chronological implementation journey, focusing on how we identified and fixed issues. Key subsections:

- **4.1 Implementation Journey Overview**: Introduce that the implementation went through multiple versions, paralleling the design evolution. State that we will describe the critical implementation details of **v6.0, v7.0, and notable experiments v8.0** (and mention v9.0 if relevant, though v9 might have limited code differences beyond retraining with more classes). Explain that each version's implementation taught us something: v6.0 revealed pitfalls, v7.0 was the corrected rebuild, and v8.0 was a short-lived fork to test an assumption [77] [78].

- **4.2 Data Acquisition and Preparation**: Describe how we built our datasets:

- Early on, we had a small hand-labeled dataset (~5k papers) for v6.0 [79] [80]. This was aggregated from various sources (mention e.g. arXiv API for CS, manual collection for IS/IT as noted in logs where CS v2.4 came from arXiv [81]). Show the class distribution issue: e.g., IT only ~506 samples in that set (9%) [80] [82]. We can include Table 4.1 from the draft showing discipline counts in the v6.0 dataset [83] [82] to highlight the imbalance.
- Implementation step: how we expanded to **Master.csv (26,944 papers)** for v7.0 [84] [85]. This is a major implementation undertaking – mention that we pulled in additional data sources to increase coverage (perhaps we combined previous sets or fetched more from databases). The key is to note the five-fold increase in data and how we integrated it. Cite the code snippet showing the loading of Master.csv and its shape (26944, with columns title, abstract, discipline, subfield, methodology) [86] [87]. This demonstrates the new dataset's scale and that each paper has all three labels.
- Preprocessing implementation: Summarize how text preprocessing was implemented (as a function cleaning punctuation, lowercasing, removing emails/URLs, etc.) [88] [89]. Mention any challenges (like dealing with special characters, ensuring consistent processing across the dataset) and confirm that the same preprocessing was applied consistently, thanks to using a unified pipeline in v7.0.
- Splitting data: Highlight the implementation of the **train/validation/test split** using stratified splitting. In code terms, we called `train_test_split` twice to get a 64/16/20 (train/val/test) split with discipline stratification [90] [91]. Emphasize that this was done *before* augmentation or vectorization (as a correction to v6.0's flawed approach) [92] [93]. We can show a brief code snippet (as in the draft) illustrating this proper splitting procedure [94] [95].

- Data augmentation: Explain the implementation of augmentation differences. For v6.0, augmentation was applied globally (leading to leakage); for v7.0, we implemented augmentation only on the training set portion. If possible, mention the library (e.g., **nlpaug** for synonym replacement as used in v4/v5) and how we parameterized it (v5 used 30% synonym replacement rate) [96] [97], whereas v7.0 used a conservative 50% training augmentation (meaning each original training sample had at most one augmented variant, preserving authenticity) [63] [60]. This shows the changes in *implementation details* to enforce design decisions.

- **4.3 Model Implementation Details**:

- **v6.0 Implementation**: Describe how we implemented the multi-feature ensemble. E.g., in code, we created several `TfidfVectorizer` instances for different feature sets (show a snippet defining them with various n-gram ranges and parameters) [46] [98]. Mention we combined their outputs horizontally with domain-specific features (list a few examples of domain features, e.g., counts of keywords like "et al.", presence of certain terms per discipline, etc., if logged – the log mentions 27 domain features in one snippet [99] though in v6 final it was 46 domain features [50]). Then, we trained an ensemble of XGBoost models: show how the code looped to train 7 models with different seeds and subsample rates [100] [48]. Indicate that we saved all these models and also the best single model. Cite the artifacts list to illustrate what was produced: `xgb_final_model_v6.0.pkl`, `ensemble_models_v6.0.pkl`, etc [101] [102]. This gives evidence of robust implementation.
    - Also note any implementation of *label encoding* or pipeline encapsulation. E.g., we created a `DisciplineClassifierV6` class (as seen in code) to organize the process [46] [100]. This object-oriented approach in notebooks made it easier to track experiments.
    - **Data leakage in implementation**: Very important to mention how we discovered the leak. In practice, we noticed something was off when deploying v6.0 (the need for manual corrections). We then scrutinized the code and saw that e.g. `TfidfVectorizer.fit_transform` was called on the entire dataset at once in v6.0 notebooks, and augmentation was done on `df` before splitting. We can reference the log summary of these mistakes [28] [103]. Possibly include a brief snippet or description of the offending code (for instance, loading all data, augmenting it, then splitting, which we can explain in words without showing too much code). This sets up how we implemented the fixes next.

- **v7.0 Implementation**: Explain the reimplementation process. We essentially rewrote the pipeline from scratch (or refactored heavily) to ensure correctness. Highlight key implementation points:

    - We used a **shared random seed and index tracking** to ensure that all classifiers (discipline, each subfield, methodology) used the exact same train/val/test split indices from Master.csv [60] [76]. The log mentions a `split_indices_v7.pkl` artifact to enforce this consistency [60] [76]. Implementation-wise, we likely generated indices once and reused them – mention that approach.
    - We created an `OptimizedFeatureExtractor` class for v7.0 (as seen in code) that encapsulates two TF-IDF vectorizers (with fixed max_features=2000 and 1000, etc.) [52] [75]. This class fits on training data only and transforms train/val/test separately. We should describe this and perhaps include a snippet where we fit the vectorizer on `X_train` and transform `X_val`/`X_test` with it, to explicitly show how we avoided leakage.
    - Training the models: Show that for v7.0 we stuck to a **single XGBoost model** per classification task, tuned through cross-validation. We can reference code where we used `GridSearchCV` or at least did manual hyperparameter tuning on validation. (The draft's design section snippet shows an example of GridSearchCV usage from v2.4, which we could contrast with v7's simpler approach or mention we still did some parameter tuning but within the training set context) [104] [105].
    - Cross-validation: We integrated cross-val in the implementation. The evaluation code snippet shows using `cross_val_score` on the training data [106]. We can say that in implementation, we ensured to evaluate our models with 5-fold CV during training to verify stability, which was not done in v6.0 (so no code for it in v6.0 notebooks).
    - Pipeline integration: Mention that we developed a **unified pipeline script** or notebook (`unified_pipeline_v7.ipynb`) that could take an input and run all classifiers in

sequence, applying the trained models. In code, this meant loading the three models and writing logic to pass the output of discipline to select subfield model, etc. If code exists, we might not show it due to space, but we describe how the final Jupyter notebooks allow end-to-end classification of a new paper. We also added conveniences like confidence thresholds (if discipline confidence < X, maybe flag uncertain) – the logs mention "confidence-based routing and correction" in unified pipeline [107]. Implementation-wise, that might be simple `if` statements checking model probabilities and overriding when needed; we can mention that conceptually.

- **v8.0 Implementation (Baseline Verification Experiment)**: Even though v8.0 was not a long-lived production version, describing how we implemented it will show thoroughness. We took the methodology classifier and split it into two models again, similar to v2.6. Note how we reused code: possibly we had a notebook for v8.0 where we trained a binary classifier for Mixed vs Not, and another for Qual vs Quant on the non-mixed subset. We should mention how easy it was to spin this up given our modular pipeline (just training new models on the same Master.csv labels but using a different labeling scheme derived from the original – Mixed vs others). This demonstrates adaptability of our implementation. Also note the results: v8.0 achieved ~82.76% accuracy on methodology with that approach [108]. We'll detail the outcome in Evaluation, but here we can say the implementation confirmed it was straightforward to try but ultimately did not replace the main model.

- **v9.0 Implementation (Detailed Methodology Classifier)**: Describe how we implemented the 8-category methodology classifier. We had to create a new dataset variant (`MASTER_DETAILED_METHODOLOGY.csv`) labeling each of the 26,944 papers with one of 8 method categories [109]. Mention that implementing v9.0 mostly meant reusing the v7 pipeline code on this new label set – a testament to our code's flexibility. We changed the label encoder to these 8 classes and perhaps adjusted some augmentation (since more classes, some smaller, maybe we ensured not to overfit them). The result was a model (XGBoost) with about 78.4% accuracy and no methodological flaws [68] [69]. Cite the log to show performance and that it was properly validated (cross-val ~77.1%) [68]. We should note that implementing this did not require new model architectures, just more classes and careful analysis, which we document in the Conclusion.

- **4.4 Validation & Testing**: Explain how we tested and validated the implementations:

- During development, we maintained logs and metrics. Mention that after training each model, we computed classification reports and confusion matrices using scikit-learn (the draft shows code for that) [110] [111]. We saved these outputs (noting the metrics files in artifacts like `metrics_v7.txt` for each model) [112] [113]. This shows good software practice.

- We also did manual sanity checks. For example, after v7.0, we likely fed a few known abstracts to the pipeline to see if outputs made sense (the logs mention adding sample papers with known categories to the demo notebook) [114]. In the report, we could mention an anecdote: *"For instance, a* Blockchain in Supply Chain *paper was correctly classified as IT discipline, Emerging Tech subfield, Qualitative method in our streamlined classifier, aligning with expectations."* Such an example (from logs: "Blockchain-based Supply Chain – IT/EMERGING/QUAL" was used as a sample) [115] gives a concrete feel to the implementation's success.

- **4.5 Important Notebooks and Reproducibility**: List the final and key notebooks in the repository, indicating their purpose:

- `Notebooks/current/` – contains the **v7.0 production notebooks** for each classifier (discipline_v7.ipynb, etc.), which are the primary implementation artifacts [116].
- `Notebooks/unified/` – contains unified pipeline and possibly the integrated demo notebook (for example, a notebook that runs all classifiers in sequence on new inputs) [117] [118].
- `Notebooks/v6.0_educational/` – holds the frozen v6.0 notebooks illustrating the flawed approach (for reference/education) [119].
- `Notebooks/v8.0_educational/` – contains the false premise methodology experiment notebooks [119].
- `Notebooks/v9.0_educational/` – contains the 8-category methodology classifier development notebooks [119].
- `Notebooks/legacy/` – all v1–v5 historical notebooks (we can mention these exist for completeness, documenting early implementation trials) [120].
- Emphasize that each is documented and we structured the repository for clarity and replication (the logs note that all major directories have README documentation as of final wrap-up) [121] [122].
- Also mention the **LOGS.md** (development log) and how it was used to track implementation progress daily [123], underscoring our commitment to transparency. This ties into reproducibility: anyone can follow the logs and notebooks to reproduce results.

(The draft implementation chapter already includes lots of code and explanation, especially for dataset and pipeline differences [79] [86]. We will trim any overlapping narrative that is covered in Design or Evaluation to stay within 2000 words. We should focus on how *we implemented (code and decisions), while referring to results only as needed to illustrate a point. We'll ensure the data expansion (5k to 26k) is clearly described, as it's a major implementation achievement [84] [86]. We'll also integrate v8.0 and v9.0 implementation notes to demonstrate full breadth of work, though briefly. If word count is tight, we might shorten detailed code listings and instead describe them in text or pseudo-code.)*

Chapter 5: Evaluation (≤2500 words)

In this chapter, we evaluate the performance of our classifiers and, critically, analyze the difference between flawed and sound methodologies. We also validate that the final system meets the objectives. The structure will cover quantitative results, error analysis, and how experiments like v8.0 and v9.0 informed our evaluation.

- **5.1 Evaluation Approach**: Describe the evaluation protocol used for all models:
- Mention that we set aside a test set (20% of data, ~5389 papers) that was never seen during training or model selection [60] [76]. All accuracy and F1 results we quote for v7.0 are on this held-out set.
- We also used 5-fold cross-validation on training data to get more robust performance estimates and standard deviations [55] [124]. Cite that the cross-val for discipline v7 was ~91.73% ±0.94% [125] [126], which closely matched the test accuracy ~92.3%, indicating no overfitting.

- Contrast with v6.0: it did not use a proper validation set or cross-val (point out as a flaw) [127] [29], which is why it's evaluation was misleading.

- **5.2 Metrics Used**: Provide a brief overview of the metrics we consider: Accuracy, Precision, Recall, F1 (macro and weighted) [128] [129]. Possibly include a small table defining them (like Table 5.1 in draft) to show thoroughness [130] [129]. Emphasize that **macro F1** is important since it averages over classes and we care about minority class performance (e.g., "Mixed" methodology or the IT discipline which had fewer samples). We likely prioritized macro F1 and per-class F1 in

our analysis to ensure the model works for all classes, not just achieving high overall accuracy by dominating on the large classes.

- **5.3 Discipline Classifier Results**:

- **v6.0 (Flawed) vs v7.0 (Final)**: Present the side-by-side results for discipline classification. The draft has Code Snippet outputs and confusion matrices:

    - v6.0: 94.77% accuracy, macro F1≈0.948 [110] [131] . Highlight how suspiciously high these were, and that per-class accuracy was extremely high for all classes (even IT ~97%) [131] . This initially looked great.
    - v7.0: 92.32% accuracy, macro F1≈0.85 [132] [133] . Show the per-class breakdown: CS ~95% F1, IS ~93%, IT ~66% F1 (from the classification report) [132] [134] . Note the drop in IT performance (IT accuracy went from 97% in v6 to ~61% in v7 test) – this is likely because v6 overfit the few IT samples, whereas v7 gives a more realistic result [135] [134] . This difference is a concrete manifestation of overfitting vs generalization.
    - Include a confusion matrix for v7.0 discipline (the draft had a Figure 28 showing one) to illustrate where misclassifications occur (likely some IT papers get misclassified as IS or CS, explaining the lower IT recall) [134] [136] .
    - **Analysis**: Explain that the small drop (2.45% absolute) in accuracy from v6 to v7 is actually a positive sign – v7's performance is *honest*. We can say: *"v6.0's nearly 95% accuracy was an illusion from data leakage; v7.0's ~92% is the true performance, as confirmed by consistent cross-val and held-out tests* [137] [138] *."* We can cite the draft line emphasizing this key finding: that 92.3% properly validated is more valuable than 94.8% with flaws [139] [137] .
    - Error analysis: We should interpret the v7 confusion matrix: e.g., many errors involve IT papers being classified as IS (perhaps because IT abstracts might resemble IS in terminology). This reveals a challenge: the IT class had the least data and is conceptually close to IS, hence lower performance. We'll mention this and perhaps give an example if we have one (if not, a hypothetical: "An IT paper on network administration might be mistaken for an IS paper on IT governance due to similar terminology"). Acknowledge this limitation, which could be improved with more IT data or specialized features (note: future work or mention that we augmented data, but IT still lagged).
    - Also note that the discipline classifier meets the target of ~85%+ accuracy with comfortable margin, so it's successful.

- **5.4 Subfield Classifiers Results**:

- Give results for each discipline's subfield classification under the final system (v7.0). The log shows:
    - CS subfield ~82.9% accuracy [140] ,
    - IS subfield ~80.3%,
    - IT subfield ~85.4% [140] .
    - These are the numbers on test sets presumably. We should present these and perhaps list the macro F1 if available.
- We should note that subfield classification is inherently harder (more classes: CS had 6 subfields, etc.). The results around 80-85% are reasonable and within our target range (80-85% as stated in project goals [141]).
- If possible, provide a brief error analysis: e.g., *In CS subfield, confusion often occurred between AI/ML and CV for papers about machine learning in computer vision – because those topics overlap.* Or mention the **AI/ML disambiguation** issue: originally we needed a separate model to split AI vs ML in v2.4 [81] [142] , but in v7's taxonomy we treated "AI/ML" as one category to avoid that

confusion. This appears to have been effective, as indicated by a decent F1 for that category (if known).

- The draft or logs provide a detailed per-class metrics table for subfields (the logs [20] include a table of precision/recall for each subfield under v6.0 unified pipeline [143] [144], but not for v7.0). We may not need to replicate a full table in the report; instead, summarize performance and note any particularly high or low subfield performance. For example: "In CS, **Security (SEC)** had the highest F1 (~0.90), while **NLP** slightly lower (~0.88), reflecting data distribution differences. In IS subfields, performance was more uniform ~0.82 F1 across categories. In IT, **CloudIT** and **DevOps** saw ~0.87 F1, whereas **Emerging Tech** slightly less – likely due to the heterogeneity of what counts as 'emerging'." – These insights show we examined which subfields the classifier finds easier or harder.
- **Legacy comparison**: If we have any baseline for subfields (like v2.4 results), we can briefly mention them to show improvement. For instance, v2.4 CS subfield was ~75% [39], so 82.9% in v7 is a good improvement. IS subfield was already high in v2 (89%), but that was on a tiny curated set [39], whereas 80% on 10k IS papers in v7 is more trustworthy. We likely won't dive deep here due to space, but a sentence noting we significantly expanded and still got strong performance is worthwhile.

- Confirm we achieved the goal for subfields (target was ~80-85% as per proposal, and we did).

- **5.5 Methodology Classifier Results**:

- **v6.0 vs v7.0**: Outline how methodology classification improved in quality from v6 to v7:

    - v6.0 methodology had an inflated accuracy of 91.87% (mentioned in intro table) but this was using an ensemble that likely leaked data as well [145]. It also reported a very low Mixed F1 in earlier versions which led us to think Mixed was a problem (e.g., in v2.5 Mixed F1 was 0.20 [146] [147], and even if v6.0 got higher overall accuracy, if leakage was present, its Mixed F1 might have been artificially improved or not truly tested).
    - v7.0 methodology achieved 86.92% accuracy on the test set [148]. More importantly, Mixed F1 = 0.68 (which is dramatically higher than the ~0.20–0.35 range from earlier models) [148] [149]. Qualitative and Quantitative F1s remained high (~0.82–0.83) [148] [149]. This indicates the final model can reliably detect Mixed method papers about 68% of the time F1-wise – a significant achievement given how subtle Mixed methods can be (Mixed papers share traits of both Qual and Quant, making them hard to identify).
    - Show a mini-table of per-class metrics:
    - Qual F1 ~0.82, Quant F1 ~0.80, Mixed F1 ~0.68 in v7.0 [149] [150].
    - Compare with an earlier snapshot like v2.3: Qual 0.83, Quant 0.81, Mixed 0.35 [146] [147] – to highlight how Mixed improved (0.35 → 0.68) while Qual/Quant stayed high [151] [152]. We can cite the log/draft snippet documenting v2.3 vs v2.5 Mixed F1 drop [153] [154] and then state v7.0 fixed this [155] [156].
    - **Analysis**: Explain what contributed to this improvement: the much larger training data and better validation likely allowed the model to learn what truly distinguishes Mixed-method abstracts (perhaps the presence of both quantitative results and qualitative discussion terms). Also, training on 26k papers probably means we had many more Mixed examples to learn from than the earlier ~2000-paper set [157]. Essentially, we solved the "Mixed problem" not by fancy algorithms, but by *data* and *methodology* improvements.
    - We should also mention that 86.9% accuracy met our goal (>85%) for methodology.

- **v8.0 evaluation**: Now bring in the results of the v8.0 two-stage methodology classifier to illustrate why it was ultimately shelved. The false premise was that v7's Mixed F1 was only ~0.35,

but in reality it was 0.68 [158] [159]. So v8.0's new model got Mixed F1 = 0.564 with 82.76% accuracy [160] [108], which is actually worse than v7.0 (Mixed F1 0.68). We will cite the log that explicitly compares these: *v8.0: Mixed F1 0.564 vs v7.0's 0.68* [160] [108]. This comparison in our evaluation section drives home the point that more complex approach (two-stage) did not help because the baseline (v7) was not weak to begin with.

    - We'll interpret this: v8.0 likely overfitted or added error by splitting the task – perhaps the binary Mixed classifier misclassified some Mixed papers as non-mixed, and then those got mis-labeled in second stage, etc. Also, adding another threshold could introduce hyperparameter sensitivity (we had to pick 0.15 threshold in v2.6) [64] [67].
    - The educational value: We verify that you must **always validate assumptions**. If we had re-evaluated v7 properly (which we did), we would know Mixed F1 was fine, avoiding the need for v8.0. So in evaluation, mention: *"This experiment (v8.0) taught that reacting to a perceived weakness without thorough validation can lead to unnecessary complexity. Proper re-evaluation showed v7.0 was strong, so v8.0 did not provide real gain* [65] [66]*."*
    - This addresses the committee's concern by showing we *did* attempt a more sophisticated classifier and we critically evaluated it, reinforcing our final choice.

- **v9.0 evaluation**: Finally, discuss the outcome of the 8-category methodology classifier experiment:

    - Accuracy ~78.4%, cross-val ~77.1% [68]. Clearly lower than the 86.9% on 3-category, which is expected because 8 classes is a harder task.
    - More importantly, discuss the **qualitative evaluation** of v9.0: The log details several issues – categories overlapping (e.g., ALGO vs SYST), annotator inconsistency, reduced practical value [71] [16]. In evaluation, we should say: *"Despite correct methodology and reasonable accuracy, the 8-class model was difficult to interpret and use: many papers legitimately belonged to multiple of the new categories, making the classification less reliable. For instance, a paper might be both algorithmic and experimental – v9.0 would force a single label (ALGO vs EXPT) leading to confusion."*
    - We will cite the log points such as *"Categories not mutually exclusive (ALGO vs SYST, CASE vs EXPT)"* [161] and *"annotation challenges led to inconsistent labeling"* [161] to support this analysis. Also note that human annotators (in this case, ourselves) had trouble cleanly assigning one of 8 categories to some papers – which is an evaluation from a usability standpoint.
    - Summarize that *while v9.0 was technically sound and taught us about advanced augmentation (which kept performance decent despite more classes)* [68] [162]*, the evaluation outcome was that the complexity it introduced was not justified by real-world needs.* We decided its granularity wasn't practically useful (most users wouldn't need such fine distinctions, and consistency was low). Therefore, we **decided to revert to the simpler 3-class methodology scheme** in production. This is a critical evaluation-driven decision: it highlights engineering judgment beyond pure metrics [163] [164].
    - By including this, we address the concern about "evolution beyond 3 methodologies" – we show we tried it, evaluated it thoroughly, and consciously chose not to adopt it due to sound reasons. This demonstrates both originality (we attempted a novel idea) and critical thinking (we didn't blindly chase complexity; we assessed its real value).

- **5.6 Additional Evaluation – Error Analysis & Case Studies**:

- **Error examples**: If space allows, we can present one or two concrete misclassification examples and discuss them:
    - E.g., find a case where the classifier made a mistake and explain why (the draft feedback wanted some qualitative insight into why some categories are hard) [165] . Perhaps *Mixed methodology misclassifications*: in v7, 32% of Mixed papers were still misclassified (since recall ~0.61). We can hypothesize that those misclassified Mixed papers might have very subtle cues or shorter abstracts that focus on one aspect. If we have a specific example from our validation set, we could describe it (like: "Paper titled *'A Study on Agile Practices in Cloud Projects'* was actually Mixed (contained both qualitative interviews and quantitative issue tracking analysis) but our model predicted Quantitative, likely because the abstract emphasized a numerical survey result and underplayed the qualitative part."). This kind of discussion shows deep engagement with the results.
    - **Confusion matrix insights**: We should mention any notable confusion matrix patterns across all tasks. For methodology, confusion matrix might show that most errors for Mixed are being predicted as Quantitative (maybe the model leans quantitative if unsure). That tells us Mixed papers tend to have at least some quantitative signal.
    - For discipline, confusion likely mostly between IS and IT, as discussed (because CS vs others is easier due to vocabulary differences).
    - For subfields, perhaps confusion within each domain's subfields (like CS: AI/ML vs Data Science (if present), or IT: Cloud vs Emerging, etc.). If we have those details we can mention one.
- **Confidence analysis**: The unified pipeline gave us confidence scores. We can mention how we analyzed confidence distributions as part of evaluation. The log snippet in future work shows an example where mean confidence for methodology was lower (0.66) than discipline (0.90) [166] [167] , meaning the model is less sure about methodology predictions – which aligns with intuition that methodology classification is harder. We can state: *"Our system also provides a confidence score for each prediction. We observed that methodology classifications had lower average confidence (around 0.66) than discipline classifications (~0.90)* [166] [167] *, indicating the model itself recognizes the greater ambiguity in identifying research methodology. This can be used to flag low-confidence cases for human review."* This ties into the robust evaluation and potential use of confidence in practice.

- **Validation of no-leak**: We should note that we double-checked v7.0's results for any hint of leakage by ensuring performance on validation and test were consistent, and by verifying that augmentation did not slip into test. Essentially, affirm the integrity of evaluation (the presence of cross-val and separate test is evidence) [90] [92] . Perhaps mention we even ran the v7 models on some new papers (post-project papers or from different sources) to see if they behave reasonably – extending evaluation to a small "real-world simulation." If we did any such test (maybe picking a few random new abstracts), we can say they seemed plausible. This reinforces reliability.

- **5.7 Summary of Evaluation**: Conclude the evaluation chapter by summarizing:

- All three classification levels meet or exceed the required performance for practical use (92.3% discipline, ~83% subfield, 86.9% methodology) [168] [140] . These numbers should be highlighted as the final outcome, perhaps in a short paragraph.
- The critical comparison of v6.0 vs v7.0 shows the *importance of proper methodology*: explicitly state: *"Our evaluation clearly demonstrated that v7.0's slightly lower scores are far more trustworthy* [139] *. The 2–5% performance drop in each classifier (compared to v6.0) eliminated the need for manual result fixes and generalizes much better* [169] [170] *."*

- We also gleaned that attempts to further increase complexity (v8, v9) did not yield better practical outcomes, reinforcing the choice of the v7.0 approach. This is a major evaluative conclusion – essentially validating the project's final direction.

(The evaluation chapter in the draft is quite detailed, and we will preserve those strong analyses [171] [172]. We will ensure to incorporate explicit mention of v8.0 and v9.0 evaluations since the draft mainly focused on v6 vs v7. By doing so, we fully address the feedback: we show we did consider more complex paradigms and evaluated them. We also might need to trim some code or metrics tables if word count is heavy – focusing instead on interpretation. But we should keep key tables/figures like the comparative methodology F1s, the data flow figure showing flawed vs proper evaluation process [173], etc., as those directly support our points. We'll also add at least a small section on ethical/fairness evaluation if not here then in conclusion, since feedback mentioned bias – however, we plan to put detailed ethical considerations in conclusion section 6.4.4 as the draft does [174] [175].)

Chapter 6: Conclusion (≤1000 words)

The conclusion will tie everything together, highlight the project's contributions, and reflect on both technical and broader issues. We will structure it into subsections as in the draft (which already has a comprehensive conclusion). Key points to cover and expand/adjust:

- **6.1 Summary of Achievements**: Summarize the final state of the project:
- Reiterate that we built a *comprehensive hierarchical classifier* for academic computing papers that achieves high accuracy at all levels [176] [177]. We can list the final metrics succinctly here (92.32% discipline, ~82–85% subfields, 86.92% methodology) [176] [178]. This confirms the project met its primary objectives.
- Emphasize the scale: we ended up training on 26,944 papers covering 3 disciplines, 15 subfields, 3 methodologies – and thus the classifier handles 135 category combinations (as mentioned earlier) reliably [36] [37].
- Mention how each component performs well: possibly highlight that discipline and subfield classifiers are very strong for their tasks, and the methodology classifier – which was initially the hardest – now has acceptable performance (improved Mixed detection). We could say the Mixed F1 of 0.68 is a notable success, directly addressing the earlier weakness [179] [180].

- Optionally include a final architecture/performance summary diagram (the draft had Figure 30) which provides an at-a-glance view of the final system and its metrics [181] [182]. This reinforces the achievement in a visual way.

- **6.2 Reflection on Methodological Lessons**: This is the heart of the conclusion – stressing that the biggest contributions are the lessons learned about ML methodology:

- Reiterate the **v6.0 vs v7.0 story** in brief: how chasing metrics led to flawed v6.0, and how careful validation in v7.0 revealed truth [183] [184]. State that this project demonstrates the critical importance of proper train/test separation, reasonable feature sizes, and having a validation set – these are often mentioned in textbooks, but our work provides a real case study confirming their necessity [28] [29].
- Mention specific pitfalls identified:
    - Data leakage (augmentation before split, vectorizer fit on full data) – and how we fixed it [28] [103].
    - Overfitting due to too many features vs too little data – v6 had ~11546 features for 8128 training samples [49] [50], v7 had 3000 for ~22k samples [185] [186]. This massive reduction improved generalization as evidenced by feature-to-sample ratio improvement.

    - Lack of validation in v6 masked issues; v7's use of validation & cross-val exposed the model's true performance early.

- We can include a summary comparison table (like Table 6.1 in draft) of v6.0 vs v7.0 methodology differences and their impact [187] [188]. That table clearly lists augmentation timing, feature count, training size, validation presence, cross-val – showing v7.0's approach in each case and how it resolves v6.0's flaw [188] [189]. This is a great concise illustration of lessons learned.

- **6.3 Addressing Initial Concerns**: This section directly responds to the feedback and demonstrates the project's depth:

- **Complexity of Problem**: Stress that although at a glance the project might seem to classify only "3 paradigms," in reality it tackled a much more complex hierarchical classification problem (as we've noted, 135 combinations) [37]. So the concept *was not too simple* – it involved nuanced distinctions at multiple levels. We should explicitly mention this to address the reviewer's comment about simplicity: *"While the methodology classification uses three top-level categories, the overall classification problem spans 135 fine-grained classes when considering all hierarchy combinations [37]. This inherent complexity required careful design and was anything but trivial."*
- **Methodological Sophistication**: Highlight that the project's sophistication lies in methodology and validation techniques, not just fancy models [190] [191]. We can say we prioritized a sound experimental framework, which is a more subtle but ultimately more significant form of complexity (the feedback was concerned we only did something simple; we counter that by showing we dealt with advanced ML methodology issues – which is a high-level complexity of its own).
- **Evolution beyond 3 methodologies**: Here we explicitly mention the v9.0 experiment. Acknowledge the initial feedback that only identifying 3 research paradigms might be too simple [192] [193], and demonstrate we took it seriously by implementing an 8-category methodology classifier (v9.0). Summarize what happened: we achieved a working model for 8 categories [68], but found that those categories overlapped and were hard to use consistently [71]. Thus, we provide a reasoned argument that sticking to 3 broader categories is actually a *better choice* for this domain – a conclusion backed by our empirical experiment. This shows critical thinking and ensures the reader knows we didn't ignore the suggestion; we tested it and have evidence for our decision.
    - We will phrase it like: *"In response to concerns about the simplicity of using only three methodology categories, we prototyped a more fine-grained 8-category methodology classifier (v9.0). The model was properly trained (78.42% accuracy) [68], but evaluation revealed substantial conceptual overlaps and annotation inconsistencies [71]. This experiment reinforced that the original three-category scheme (Qual/Quant/Mixed) struck the right balance between clarity and detail [17]. The attempt at greater granularity provided valuable insight: sometimes simpler is better in practice – an unexpected but important outcome."* (This aligns with the log's note that v9.0 exemplified *"Simple Often Beats Complex"* as a key lesson [194].)

- **Model Complexity vs Practicality**: Address the model complexity aspect of feedback. We should assert that we did experiment with complex models (transformer fine-tuning, huge ensembles) but found that they did not yield practical improvements. Instead, a simpler XGBoost approach with sound practices outperformed those complex models when evaluated fairly. This underscores an original insight: more complex architecture ≠ better results if methodology is flawed. We might reference how SciBERT+LoRA ultimately wasn't part of final, because XGBoost with proper validation did better [195] [44]. And mention how the ensemble of 7 XGBoosts (v6) was

slimmed to 1 XGBoost (v7) with only a minor performance trade-off but big gains in simplicity and reliability [58] [59] .

    - Essentially, we frame this as a positive: *the project demonstrated technical depth by exploring advanced models, but also the maturity to choose a simpler solution when it proved more effective.*
    - We can cite the log or draft where it states v6.0's complexity vs v7.0's simplicity (the design Figure 9 or the conclusion notes) to support this: e.g., *"v6.0 used multi-model ensembles and 11k+ features, whereas v7.0 used one model and 3k features, yet achieved nearly the same accuracy [45] [53] . This project evidences that judicious feature reduction and proper validation can outperform brute-force complexity."*

- **6.4 Broader Impacts and Contributions**:

- **Educational Contribution**: Summarize that we documented every step (LOGS.md, educational notebooks) so others can learn from both our successes and mistakes [196] [197] . Highlight that preserving the flawed v6.0 and misguided v8.0 as case studies is intentional [198] [158] – it provides a rare detailed look at how *not* to do it, alongside how to do it correctly. This is a novel contribution to the academic community's understanding of ML praxis.
- **Practical Guidelines**: We can list a few concrete guidelines that emerged (the draft conclusion enumerates some) [199] [200] :
    1. Always split data before augmentation or feature selection [54] [55] .
    2. Keep feature count in check relative to sample size (we even gave a heuristic: e.g., at least 5–10 samples per feature) [201] [202] .
    3. Use cross-validation and a separate test for reliable evaluation [29] .
    4. Be cautious of too-granular categorization without clear definitions (lesson from v9.0).
    5. etc. These can be presented as a short list in the conclusion, showing how our work yields general advice.
- **Reproducibility & Transparency**: Note that all code, data (Master.csv), and models are openly available, and we structured the repository for clarity [203] [204] . This is important for a scholarly report – it shows we adhere to best practices by enabling others to replicate or build on our work. We can mention that we even included extensive documentation (Readmes for every major version in the repo) [121] [122] , ensuring that the project is accessible.

- **Ethical Considerations**: Address any ethical or societal considerations, as the feedback suggested. The draft conclusion has a rich section on this [174] [205] . Summarize key points:

    - *Bias*: Our dataset is mostly English and from Western sources, so the classifier might not generalize to non-English papers or might underrepresent perspectives outside mainstream conferences [206] [205] . We acknowledge this and recommend future work to expand to diverse languages/corpora.
    - *Static Taxonomy vs Evolving Research*: If we lock in these categories, there's a risk of misclassification for novel interdisciplinary work (like an AI + Biology paper) that doesn't fit neatly [207] [208] . We note that continuous updates to the taxonomy and model retraining would be needed to stay current, and that our system is a tool to aid, not replace, human curation of knowledge.
    - *Use of the System*: Clarify it's meant for organizing and searching literature, **not** for evaluating researchers or papers. Automated classifications should not be used in isolation for academic assessment (to avoid negative impacts on, say, tenure decisions) [209] [210] . We built it as a convenience tool.

    - *Transparency*: By making even the flawed models available, we allow others to inspect how errors happen, potentially catching biases or issues we didn't see [211] [212]. This transparency is part of the project's ethical approach.
    - *Environmental Impact*: Training ML models has a carbon footprint. We note that simplifying the model (v7 vs v6) had the side benefit of reducing computational cost by ~75% (since 3k vs 11k features means less training time) [213] [214], illustrating how sound methodology can align with efficiency and sustainability. This section shows we have thought about the broader implications, addressing the feedback about bias/fairness.

- **6.5 Future Work**: Outline a few directions building on this project (the draft has them):

- Adopting more advanced models *with* proper validation: e.g., try fine-tuning BERT or newer transformer on our data now that we have best practices in place, to see if it can beat 92% without leakage [215]. We note we avoided that in v7.0 to eliminate complexity, but it could be revisited in a controlled way (maybe a v10, in spirit).
- Deploying the system in real environments: integrate with a web application or API for researchers (some work on a Flask app was hinted as future in README) [216] [217].
- Expanding to other domains: our methodology can be applied to other fields (e.g., biomedical papers with their own disciplines/subfields/methods) [218] [219]. Also, making it multilingual (non-English data) as mentioned in ethics.
- Developing a standardized **validation framework** for similar projects (so that others don't repeat v6.0 mistakes) [220]. Perhaps we could publish a short methodology paper or checklist out of this.

- Continue the educational aspect: incorporate these case studies into ML curriculum (we effectively created a rich example of pitfalls and fixes).

- **6.6 Final Reflections**: Conclude with a strong closing paragraph:

- Reiterate the central message: *"The journey is as important as the destination in ML projects."* Our project's biggest success is not just the working classifier, but the demonstration that **methodological rigor and critical evaluation** are key to true success [221] [222].
- Mention that the project evolved the definition of success: initially it was "get 95% accuracy," but it became "achieve reliable performance and new understanding" [221] [223].
- State that we have contributed both a functional tool for classifying papers *and* a case study in best practices for machine learning in research applications [224] [225]. This highlights originality and depth.
- End on a forward-looking note: e.g., *"We hope this work will help future researchers build upon a solid foundation of validated methodology, applying these lessons to new challenges with confidence."* Such a conclusion leaves a positive impression of the project's significance.

(The draft conclusion already covers most of these points excellently [226] [183] [174]. Our task is to ensure we explicitly mention the response to the simplicity concern (we will add the v9.0 discussion), and ensure consistency/coherence. We should trim any repetitive parts – e.g., the introduction of the conclusion might re-list final accuracies redundantly if already in summary; we'll keep it tight. The ethical considerations and future work are well fleshed out in the draft; we'll retain those, perhaps slightly condensing if needed. By organizing into subsections, we'll avoid a wall of text and make each theme clear. Finally, we confirm the word count is under limits: the current draft was ~9400 words excluding references; with our additions (v9 mentions, etc.) we must be mindful to cut some verbosity elsewhere to stay within 9500.)

By following this structured plan for each chapter, the final report will be comprehensive and well-balanced. We will have: - Incorporated the complete chronological evolution of models (v2.3 → v9.0) within the narrative, ensuring each version's purpose and lessons are documented [157] [71] . - Identified the most mature components (v7.0 classifiers and unified pipeline) as the recommended solutions, while positioning v6.0, v8.0, v9.0 as valuable case studies rather than final solutions [227] [228] . - Addressed the review concerns by showing how the project went beyond initial simplicity, delved into complex experiments, and emerged with justified decisions grounded in evidence and critical analysis. - Ensured coherence by assigning each topic to the appropriate chapter (design vs implementation vs evaluation) to minimize redundancy. For example, the data leakage issue is described in design/implementation (what was done wrong and fixed) and analyzed in evaluation (impact on metrics), but each with a different focus. Throughout, the writing will transition logically (the end of Design sets up what Implementation will detail, Implementation findings lead into Evaluation results, etc.). - Managed word count by focusing expansion on critical analysis (e.g., discussing v9.0 in conclusion, providing error analysis) while trimming any superfluous or repetitious explanations that might exist from the draft.

Following this plan, the final report will be a thorough documentation of both the *product* (the classification system) and the *process* (the research journey), highlighting originality, technical depth, and reflective learning. Each chapter will flow into the next, collectively telling the story of how a seemingly simple idea evolved into a nuanced, rigorously validated outcome, thereby fulfilling both the project objectives and the expectations from feedback.

Sources:

- Project development log detailing version changes and lessons [157] [71]
- Repository documentation indicating final vs educational notebook organization [227] [228]
- Draft report excerpts for design, implementation, and conclusion insights [53] [183]

---

[1] [165] [192] [193] Peer Reviewed Assignments
https://www.notion.so/1f94f8a8309880a4bc71f083f1a8b719

[2] [3] [4] [5] [6] [8] [9] [10] [11] [12] [14] [15] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [30] [31] [32] [33] [34] [35] [36] [37] [45] [46] [47] [48] [49] [50] [52] [53] [65] [66] [72] [73] [74] [75] [77] [78] [79] [80] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] [98] [100] [104] [105] [106] [110] [111] [128] [129] [130] [131] [132] [133] [134] [135] [136] [137] [138] [139] [151] [152] [153] [154] [155] [156] [158] [159] [166] [167] [171] [172] [173] [174] [175] [176] [177] [178] [179] [180] [181] [182] [183] [184] [185] [186] [187] [188] [189] [190] [191] [196] [197] [198] [199] [200] [201] [202] [203] [204] [205] [206] [207] [208] [209] [210] [211] [212] [213] [214] [215] [216] [217] [218] [219] [220] [221] [222] [223] [224] [225] [226] Draft Report
https://www.notion.so/23c4f8a830988094824ae8a6aa8a652a

[7] [13] [16] [17] [28] [29] [40] [41] [42] [43] [44] [51] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [67] [68] [69] [70] [71] [76] [81] [96] [97] [99] [101] [102] [103] [107] [108] [109] [112] [113] [114] [115] [116] [117] [118] [119] [120] [121] [122] [123] [124] [125] [126] [127] [140] [141] [142] [143] [144] [145] [146] [147] [148] [149] [150] [157] [160] [161] [162] [163] [164] [168] [169] [170] [194] [195] [227] [228] LOGS.md
https://github.com/aanandprabhu30/NLP-Project/blob/8487fd6bfced66a3dd0b83958941b4c5decfcc14/Documentation/LOGS.md

[38] [39] README.md
https://github.com/aanandprabhu30/NLP-Project/blob/8487fd6bfced66a3dd0b83958941b4c5decfcc14/Notebooks/legacy/README.md