Data Generation and Methodologies Summary

1. Data Generation

1.1 Dataset Description

* \*\*Subject Area\*\*: Data Science and Machine Learning
* \*\*Complexity Level\*\*: Intermediate
* \*\*Submission Format\*\*: Jupyter Notebooks with code and analysis

1.2 Sample Characteristics

* \*\*Total Submissions\*\*: 50+ student assignments
* \*\*Submission Length\*\*: Variable (500-2000 lines of code)
* \*\*File Formats\*\*: Jupyter Notebooks (.ipynb), Python scripts (.py)
* \*\*Student Level\*\*: Graduate and undergraduate students
* \*\*Assignment Duration\*\*: 2-week development period

1.3 Evaluation Criteria

1. \*\*Code Correctness\*\* (30%): Proper implementation of algorithms
2. \*\*Data Preprocessing\*\* (20%): Appropriate data handling and preparation
3. \*\*Model Implementation\*\* (25%): Correct usage of machine learning libraries
4. \*\*Results Analysis\*\* (15%): Interpretation and discussion of results
5. \*\*Documentation Quality\*\* (10%): Code comments and markdown explanations

1.4 Data Processing Pipeline

Student Submissions → File Processing → Content Extraction → AI Evaluation → Grade Generation

1. \*\*Upload\*\*: Multiple file formats (.ipynb, .py, .txt)
2. \*\*Parsing\*\*: Extract code and text content
3. \*\*Standardization\*\*: Convert to uniform format
4. \*\*Validation\*\*: Check for completeness and readability

2. Methodologies Used

2.1 Methodology M1: Simple GPT Prompt

def get\_points\_and\_comments\_using\_GPT4(sid, student\_name, processed\_file,   
 assignment\_name, possible\_points,   
 question, rubric, temperature,   
 selected\_model, reasoning\_level):  
 prompt = f"""question:###{question}###,  
 Rubric: "{rubric}",  
 Total points: {possible\_points},   
 ``SUBMISSION TO QUESTION ABOVE.``: ###{processed\_file}###,  
 ``Verify submission against rubric included in knowledge file prior to grading.   
   
 IMPORTANT: Respond with ONLY a valid JSON object in this exact format:  
 {{  
 "points": <number>,  
 "comments": "<string>"  
 }}"""

* Single evaluation pass
* JSON output format
* Fast processing (2.3s/submission)
* Low computational overhead

2.2 Methodology M4: Chain of Thought (COT)

1. \*\*Requirement Analysis\*\*: Parse assignment objectives
2. \*\*Submission Evaluation\*\*: Code structure and functionality assessment
3. \*\*Rubric Application\*\*: Criterion-by-criterion evaluation
4. \*\*Final Assessment\*\*: Score calculation and feedback generation

def get\_completionCOT(prompt, temperature=0, selected\_model="gpt-3.5-turbo", reasoning\_level=None):  
 cot\_prompt = f"""  
 Let's think step by step about this grading task:  
   
 Step 1: Understanding the Assignment  
 [Assignment analysis]  
   
 Step 2: Evaluating the Submission  
 [Submission review]  
   
 Step 3: Applying the Rubric  
 [Criterion evaluation]  
   
 Step 4: Final Assessment  
 [Grade and feedback]  
 """

* Multi-step reasoning
* Transparent evaluation process
* Processing time: 3.5s/submission
* Enhanced accuracy and consistency

2.3 Methodology M5: Chain of Thought with Multi-Calls

def generate\_deduction\_plan(question, rubric, possible\_points):  
 prompt = f"""  
 Assignment: {question}  
 Rubric: {rubric}  
 Total Points: {possible\_points}  
   
 Create a detailed deduction plan with:  
 - Specific criteria for evaluation  
 - Point values for each criterion  
 - Deduction amounts for common issues  
 """

def evaluate\_criterion(submission, criterion, deduction\_plan):  
 for criterion in deduction\_plan.criteria:  
 evaluation = assess\_criterion(submission, criterion)  
 deductions = calculate\_deductions(evaluation)  
 comments = generate\_criterion\_feedback(evaluation)

def final\_grade\_calculation(evaluations, deduction\_plan):  
 total\_deductions = sum(evaluation.deductions for evaluation in evaluations)  
 final\_score = possible\_points - total\_deductions  
 comprehensive\_feedback = synthesize\_feedback(evaluations)

class Criterion(BaseModel):  
 description: str  
 deduction: float  
  
class DeductionPlan(BaseModel):  
 criteria: Dict[str, Criterion]  
  
class EvaluationCriterion(BaseModel):  
 deduction: float  
 comments: str  
  
class SubmissionEvaluation(BaseModel):  
 evaluation: Dict[str, EvaluationCriterion]  
 total\_deduction: float  
 final\_comments: str

* Multi-phase evaluation
* Structured deduction planning

3. Models Used

3.1 AI Models

* \*\*GPT-3.5-turbo\*\*: Baseline model for all methodologies
* \*\*GPT-4\*\*: Enhanced reasoning capabilities
* \*\*GPT-4o\*\*: Latest model with improved performance
* \*\*GPT-5\*\*: Advanced reasoning with configurable levels

4. System Architecture

4.1 Processing Engine

┌─────────────────────────────────────────────────────────────┐  
│ User Interface Layer │  
│ (Streamlit Web App) │  
└─────────────────────┬───────────────────────────────────────┘  
 │  
┌─────────────────────▼───────────────────────────────────────┐  
│ Processing Engine │  
│ ┌─────────────┐ ┌─────────────┐ ┌─────────────────────┐ │  
│ │ M1: │ │ M4: │ │ M5: │ │  
│ │ Simple GPT │ │ Chain of │ │ COT with │ │  
│ │ │ │ Thought │ │ Multi-Calls │ │  
│ └─────────────┘ └─────────────┘ └─────────────────────┘ │  
└─────────────────────┬───────────────────────────────────────┘  
 │  
┌─────────────────────▼───────────────────────────────────────┐  
│ AI Model Integration │  
│ (OpenAI GPT-3.5/4/4o/5) │  
└─────────────────────────────────────────────────────────────┘