Supplementary Materials for FMM-Agent: Evolving Feature Meta-Models for Industrial Imbalanced Scenarios via LLMs

This material provides additional implementation details for the proposed FMM-Agent framework. These materials are intended to support reproducibility and to clarify how LLM-guided feature evolution is performed in the Feature Meta-Model (FMM) space. More implementation details, including source code, configuration files, and running scripts, can be found in the open-source repository https://github.com/EMRGSZU/FMM-Agent

A. Detailed Description of Feature Meta-Models

A.1 Example of Original FMM Representation

In FMM-Agent, each feature is represented as a FMM rather than only by its raw sample values. A FMM records statistical characteristics, feature-label relevance, stability-related information, and transformation information of a feature. A simplified example of the original FMM representation used in the implementation is shown below.

```
{
    "feat_name": feat,
    "feat_type": ftype,
    "mean": mean_v,
    "std": std_v,
    "skew": sk_v,
    "kurtosis": ku_v,
    "min": min_v,
    "max": max_v,
    "q01": q01,
    "q05": q05,
    "q25": q25,
    "q50": q50,
    "q75": q75,
    "q95": q95,
    "q99": q99,
    "n_unique": n_unique,
    "missing_rate": missing_rate,
    "mi_to_y": mi,
    "info_gain": info_gain,
    "FeatureScore": 0,
    "lineage": "raw",
    "ops_chain": [],
    "gen_round": 0
```

}

This representation allows FMM-Agent to describe a feature from a statistical perspective. During evolution, crossover, mutation, and LLM-based reasoning are conducted over such structured FMMs. The final generated feature is then instantiated through an executable expression and evaluated using the proposed scoring and screening mechanism.

A.2 Motivation for FMM-based Feature Evolution

The main motivation for using FMMs is to avoid directly evolving features from noisy instance-level samples. In industrial imbalanced scenarios, raw data often contain random noise. If feature evolution is performed directly on raw instances, the generated transformations may overfit local fluctuations or amplify unstable patterns. By contrast, FMM-Agent evolves features in a statistical meta-model space, where each feature is represented by compact descriptors such as distributional statistics, relevance indicators, stability measures, and transformation rationales. This design reduces sensitivity to instance-level noise, helps the agent consider global distributional properties, and alleviates the context-window burden of LLMs because the LLM only needs to reason over structured feature summaries rather than large raw data matrices. Therefore, compared with ordinary feature engineering methods that mainly search over operator combinations in the original feature space, FMM-Agent performs a more distribution-aware and stability-oriented feature evolution process.

B. Prompt Design

B.1 Prompt A1: FMM Crossover Prompt

System prompt:
You are an expert specializing in feature engineering for imbalanced classification.
Your task: given two feature metadata JSON objects, create ONE new derived feature by combining these two parent features.
Output must be a SINGLE feature metadata object with EXACTLY the same keys as the original metadata rows:

```
{
  "feat_name": "(fAfB)", "feat_type": "...", "mean": ..., "std": ..., "skew": ..., "kurtosis": ...,
  "min": ..., "max": ..., "q01": ..., "q05": ..., "q25": ..., "q50": ..., "q75": ..., "q95": ...,
  "q99": ..., "n_unique": ..., "missing_rate": ..., "mi_to_y": ..., "info_gain": ...,
  "FeatureScore": 0, "lineage": "derived", "ops_chain": ["..."], "gen_round": <value>
}
```

Rules:
1. The new feature must be a meaningful combination of the two parent features, such as product, ratio, difference, normalized interaction, or weighted combination.
2. All statistical fields must be logically estimated from the parent metadata.
3. Quantiles must be monotonic and consistent with min/max.

4. The new FeatureScore is set to 0.
5. "mi_to_y" is the mutual information between this feature and the classification target. "info_gain" is an information gain measure for multi-class classification.
6. feat_name MUST concatenate the two parent names wrapped in parentheses, for example "(f2f4)".
7. ops_chain MUST be a list with EXACTLY ONE short natural-language string. It must state the base feature, explain how the parents are combined, summarize how mean/ std/ skew/ kurtosis/ min/ max change, and explain why the change may help minority-class discrimination.
8. gen_round MUST be max(parent.gen_round) + 1.
Return ONLY the JSON object of the new feature metadata.

User prompt:
Please perform a crossover operation on these two features to generate ONE meaningful derived feature.
Feature 1:
{feature1_json}
Feature 2:
{feature2_json}
Analyze their statistical properties, distributions, and potential relationship. Create only one crossover feature that meaningfully combines their characteristics.
Return your result in the JSON format specified in the system prompt.

B.2 Prompt A2: FMM Mutation Prompt

System prompt:
You are an AI assistant specializing in feature engineering for imbalanced classification.
Your task: given ONE feature metadata JSON object containing statistics such as mean, std, skew, kurtosis, quantiles, n_unique, missing_rate, mi_to_y, and info_gain, create ONE new derived feature by applying a meaningful mutation to the original feature.
Output must be a SINGLE feature metadata object with EXACTLY the same keys as the original metadata row:

```
{
  "feat_name": "(fAfB)", "feat_type": "...", "mean": ..., "std": ..., "skew": ..., "kurtosis": ...,
  "min": ..., "max": ..., "q01": ..., "q05": ..., "q25": ..., "q50": ..., "q75": ..., "q95": ...,
  "q99": ..., "n_unique": ..., "missing_rate": ..., "mi_to_y": ..., "info_gain": ...,
  "FeatureScore": 0, "lineage": "derived", "ops_chain": ["..."], "gen_round": <value>
}
```

Rules:
1. The new feature must be a meaningful transformation of the original feature, such as log, sqrt, clipping, scaling, tail emphasis, or ratio to its mean.
2. All statistical fields must be logically estimated from the original feature metadata.

3. Quantiles must be monotonic and consistent with min/max.
4. The new FeatureScore is set to 0.
5. "mi_to_y" is the mutual information between this feature and the classification target. "info_gain" is an information gain measure for multi-class classification.
6. feat_name MUST be different from the original feature name and clearly indicate a mutated version.
7. ops_chain MUST be a list with EXACTLY ONE short natural-language string. It must state the parent feature, describe the transformation idea, summarize how mean/ std/ skew/ kurtosis/ min/ max change, and explain why the change may help minority-class discrimination.
8. gen_round MUST be original_feature.gen_round + 1.
Return ONLY the JSON object of the new feature metadata.

User prompt:
Please perform a mutation operation on this feature to generate ONE meaningful derived feature.
Original Feature:
{feature_json}
Analyze its statistical properties, distribution, and transformation potential. Create only one mutated feature that may improve performance in an imbalanced classification setting.
Return your result in the JSON format specified in the system prompt.

B.3 Prompt A3: Refinement Prompt

System prompt:
You are an AI assistant specializing in feature engineering and statistical analysis for machine learning.
Your task: given parent and child feature metadata, analyze their statistical properties, feature scores, and operation-chain descriptions to generate trend adjustment recommendations for feature transformation.
The ops_chain contains natural-language descriptions of the transformation logic. Use those descriptions to understand the intended transformation purpose and expected statistical changes.
Return a JSON object with EXACTLY these fields:
{
  "mean": <float>, "std": <float>, "skew": <float>, "kurt": <float>, "min": <float>,
  "max": <float>
}
Each value is a trend adjustment from -1.0 to 1.0. Negative values mean decrease, positive values mean increase. Keep values moderate, usually between -0.8 and 0.8, to preserve numerical stability.
Return ONLY the JSON object with your trend recommendations.

User prompt:
Please analyze these parent and child feature metadata to generate trend adjustment recommendations.
PARENT FEATURE:
{parent_json}
CHILD FEATURE:
{child_json}
TRANSFORMATION DESCRIPTION:
{ops_description}
Compare the statistical properties, study the transformation description, and evaluate feature scores. Generate trend adjustments that transform the parent feature toward the child's statistical characteristics while maintaining numerical stability and discriminative power.
Return your recommendations in the JSON format specified in the system prompt.

C. Additional Parameter and Implementation Settings

The main text reports the core algorithmic parameters, including the number of evolution rounds m, crossover probability p_c, mutation probability p_m, feature-scoring weight λ, skewness and kurtosis decay scales c_1 and c_2, fixed-quantity screening number k, and threshold screening score τ. To further support reproducibility without overloading the main text, Table A1 summarizes additional implementation-level settings used in the reported experiments. In addition, Table A2 provides the nomenclature of the main abbreviations, variables, and symbols used in FMM-Agent.

Table A1 Additional implementation settings of FMM-Agent.

Parameter / Setting	Value / Setting	Description
Cross-validation protocol	5-fold stratified cross-validation	Used for public benchmark datasets and all compared methods under the same evaluation protocol.
Data-splitting random seed	42	Used for stratified fold construction and the random split.
Missing-value handling for FMM statistics	Non-missing values only	Distributional descriptors such as mean, standard deviation, skewness, kurtosis, and quantiles are computed after removing missing values.
Missing-value handling for MI/IG	Median imputation	Missing values are filled with the training-split median before mutual information and information gain calculation.
Continuous feature discretization	Quantile-based binning	Applied before MI and IG calculation when the number of distinct values is larger than 20.
Maximum number of bins	10	Upper bound for quantile-based discretization.

Parameter / Setting	Value / Setting	Description
Discrete-state threshold	20	Features with no more than 20 distinct values are directly treated as discrete states.
Default LLM provider	deepseek-chat	Used for FMM crossover, mutation, and fuzzy transformation planning in the reported implementation.
LLM temperature	0.1	Low-temperature setting for stable structured outputs.
Maximum LLM output tokens	4000	Default maximum response length when no provider-specific value is set.
LLM timeout	30 seconds	Request timeout in the default provider configuration.
LLM maximum retries	3	Maximum number of retries in the default provider configuration.
Python version	Python 3.11.7	Used in the reported implementation environment.
CPU requirement	Intel Core i5-12600KF	Low CPU performance requirements
GPU requirement	NVIDIA RTX 4060	Since the LLM is accessed through external APIs, local GPU computation is not required in the reported implementation.
LLM usage cost	API-dependent	Although no local GPU is required, LLM-based crossover, mutation, and fuzzy transformation planning require API calls, which may introduce monetary cost and network latency.

Table A2 Nomenclature of FMM-Agent.

Symbol / Abbreviation	Description
FMM	Feature Meta-Model
FMM-Agent	Feature Meta-Model Agent
LLM	Large Language Model
NLP	Natural Language Processing
AutoFE	Automated Feature Engineering
CoT	Chain-of-Thought
BAC	Balanced Accuracy
G-mean	Geometric mean
MI	Mutual Information
IG	Information Gain
S_i	Statistical descriptor component of FMM_i
R_i	Reasoning-chain component of FMM_i
X_i	The i-th feature
Y	Class label
$P(X_i, Y)$	Joint distribution of X_i and Y
$P(X_i),\ P(Y)$	Marginal distributions of X_i and Y

Symbol / Abbreviation	Description
$H(Y)$	Entropy of the class label
$H(Y\|X_i)$	Conditional entropy of Y and X_i
Rel_i	Feature-label relevance score
Sta_i	Distribution stability score
$Score_i$	Final score of the feature/FMM
D_i	Dispersion-related descriptor
$Skew_i$	Skewness-related descriptor
$Kurt_i$	Skewness-related descriptor

D. Discussion on Limitations and Risk Mitigation

Although FMM-Agent improves automated feature evolution in industrial imbalanced scenarios, it still has some limitations that should be noted. Since the framework uses LLMs as heuristic reasoning modules, hallucination may occur, and the generated FMMs or feature expressions may sometimes be statistically unreasonable, redundant, or non-executable. For example, an expression may contain undefined variables, division by zero, or invalid logarithmic operations. To reduce these risks, we use structured prompts, fixed input/output formats, validity checking, executable expression verification, and relevance-stability-based scoring. More importantly, LLM outputs are not directly used as final features. They are further filtered through elite screening and downstream validation.

FMM-Agent may also face difficulties when the data are extremely high-dimensional, heavily noisy, or affected by rapid concept drift. In addition, LLM calls inevitably bring extra cost, latency, and dependency on external APIs. In our implementation, these issues are partly controlled by using compact FMM representations, stable LLM calling formats, and candidate screening strategies. In future work, we will further consider stricter statistical validation, drift-aware adaptation, human-in-the-loop checking, and lightweight local model deployment to improve the reliability and efficiency of the framework.