A APPENDIX

A.1 Implementation Details

Hyperparameters. The specific hyperparameters for each module, including the RL agent, the LLM, and the Advisor^{+} mechanism. All settings were chosen to ensure a robust and fair comparison across all experiments. Table 1 lists the key settings for our LLMPipe framework. The settings were kept consistent across all datasets.

Table 1: Hyperparameter settings for the LLMPipe framework

Module	Hyperparameter	Value
	RL Agent	
Learning	Learning Rate (α)	1
	Discount Factor (γ)	0.9
Exploration	ϵ-Greedy Decay Strategy	Exponential
	Initial Epsilon (ϵ_{start})	1.0
	Final Epsilon (ϵ_{end})	0.1
	Epsilon Decay Factor	0.99
	LLM Policy Advisor	
Model	LLM Model	LLaMA-3.3-70B
	Temperature	0.1
Retrieval	Number of Retrieved Examples (k)	3
	Text Embedding Model	nomic-embed-text
	Vector Database Index	FAISS
	Adaptive Advisor Triggering (Advisor^{+})	
Triggering	Slope Threshold (θ_{slope})	0.01
	Performance Buffer Size ($\|\mathcal{B}\|$)	10 episodes
	Cooldown Period	5 episodes
	Framework Settings	
General	Max Pipeline Length ($L_{\max}$)	9
	Total Training Episodes ($E_{\max}$)	100
Random Seed	Dataset Train Test Split	0
	Seed for ϵ-greedy action selection	42, 1999, 2025, 2048, 2147, 9331, 65537

A.2 Full Experimental Results

A.2.1 Extended Comparison Baselines. To situate LLMPipe within the broadest possible competitive context, we present an extended comparison by integrating our results into the main evaluation table from CtxPipe. Table 2 includes performance data for differentiable methods (DEF, RS, DP-Fix, DP-Flex), SAGA, and other baselines alongside our LLMPipe variants. This direct comparison underscores the competitive performance of our approach.

A.2.2 Detailed Pipeline Composition. To provide deeper insight into the qualitative differences between the pipelines discovered by CtxPipe and our LLMPipe variants, we present the detailed composition of the best-performing pipeline for each method on every dataset.

Table 3 details the generated pipelines and their resulting accuracy scores. The 'Pipe Length' row at the bottom summarizes the average number of operators in the pipelines generated by each method. This table illustrates that LLMPipe not only achieves competitive or superior accuracy but often does so with different, and sometimes more complex or novel, pipeline structures.

Table 2: Extended comparison of test accuracy, including baselines from CtxPipe. Our methods (Advisor, Advisor$^+$) are added for direct comparison.

Dataset	DEF	RS	DP-Fix	DP-Flex	DL	HAI-AI	SAGA	CtxPipe	Advisor	Advisor$^+$
abalone	0.240	0.243	0.238	0.271	0.157	0.260	0.255	**0.287**	0.270	0.273
ada_prior	0.848	0.844	**0.853**	0.846	0.803	0.801	0.833	0.818	0.838	0.841
avila	0.553	0.598	0.652	0.633	0.593	0.630	0.636	0.759	**0.929**	0.751
connect-4	0.659	0.671	0.726	0.702	0.683	**0.775**	0.758	0.763	**0.775**	**0.775**
eeg	0.589	0.658	0.675	0.683	0.607	0.556	0.658	0.740	**0.839**	0.811
google	0.586	0.627	0.631	0.661	0.553	0.550	0.596	0.590	**0.675**	0.674
house	0.928	0.938	0.932	**0.952**	0.771	0.928	0.913	0.818	0.908	0.908
jungle_chess	0.668	0.669	0.680	0.687	0.717	0.760	0.745	**0.861**	**0.861**	**0.861**
micro	0.564	0.579	0.595	0.593	0.613	0.633	0.556	0.605	**0.643**	0.633
mozilla4	0.855	0.922	0.924	0.927	0.747	0.870	0.932	0.940	0.937	**0.944**
obesity	0.775	0.841	**0.891**	0.874	0.590	0.768	0.751	0.868	0.865	0.835
page-blocks	0.942	0.959	0.959	**0.973**	0.940	0.935	0.849	0.965	0.963	0.961
pbcseq	0.710	0.730	0.728	0.725	0.680	0.733	**0.866**	0.805	0.753	0.753
pol	0.884	0.879	0.903	0.916	0.873	0.916	0.888	0.949	**0.981**	**0.981**
run_or_walk	0.719	0.829	0.903	0.912	0.820	0.915	0.832	0.956	0.953	**0.976**
shuttle	0.964	0.996	0.998	0.999	0.790	0.951	0.405	**1.000**	0.987	0.986
uscensus	0.848	0.840	**0.854**	0.852	0.813	0.807	0.835	0.845	0.846	0.847
wall-robot-nav	0.697	0.872	0.905	0.913	0.927	0.896	0.841	0.946	**0.961**	0.946
AVERAGE	0.724	0.761	0.780	0.784	0.704	0.760	0.731	0.806	**0.833**	**0.820**
RANK	7.81	6.56	4.86	4.19	8.44	6.58	6.75	3.61	**3.00**	**3.19**

Table 3: Detailed composition and accuracy of pipelines generated by CtxPipe and LLMPipe variants. Numbers in brackets correspond to operator IDs in Table 2 in the main paper.

Dataset	CtxPipe		Advisor		Advisor$^+$	
	Accuracy	Pipeline	Accuracy	Pipeline	Accuracy	Pipeline
abalone	0.287	[-1, -1, 4, 24, 12, 15]	0.270	[6, 9, 11, 16, 20]	0.273	[8, 15]
ada_prior	0.818	[-1, -1, 4, -1, 6, 23]	0.838	[5, 12, 9]	0.841	[12, 6, 5]
avila	0.759	[-1, -1, -1, 24, 12, 23]	0.929	[10, 23, 18, 15]	0.751	[23]
connect-4	0.763	[-1, -1, -1, -1, 12, 23]	0.775	[15]	0.775	[15, 4]
eeg	0.740	[-1, -1, -1, 23, 24, 12]	0.839	[18, 3, 15, 9]	0.811	[10, 15]
google	0.590	[1, -1, 4, 6, 23, 24]	0.675	[2, 10, 9, 15]	0.674	[2, 11, 8, 15]
house	0.818	[1, -1, 4, 12, 23, 24]	0.908	[1]	0.908	[1]
jungle_chess	0.861	[-1, -1, -1, 24, 6, 23]	0.861	[23]	0.861	[23]
micro	0.605	[-1, -1, -1, -1, 6, 23]	0.643	[12, 15, 18]	0.633	[15]
mozilla4	0.940	[-1, -1, -1, 6, 23, 24]	0.937	[13, 10, 24, 15]	0.944	[12, 6, 23]
obesity	0.868	[-1, -1, 4, 24, 6, 23]	0.865	[10, 5, 9, 15]	0.835	[0, 15, 17]
page-blocks	0.965	[-1, -1, -1, -1, 6, 23]	0.965	[3, 10, 15]	0.961	[11, 20]
pbcseq	0.805	[1, -1, -1, -1, 6, 23]	0.753	[23, 6]	0.753	[23]
pol	0.949	[-1, -1, -1, 8, 23, 24]	0.981	[15]	0.981	[16]
run_or_walk	0.956	[-1, -1, -1, 24, 12, 15]	0.953	[12, 15, 17]	0.976	[14, 23]
shuttle	1.000	[-1, -1, -1, 23, 24, 12]	0.987	[9, 15]	0.986	[10, 5]
uscensus	0.845	[-1, -1, 4, -1, 6, 23]	0.846	[2, 5, 7]	0.847	[5, 9]
wall-robot-nav	0.946	[-1, -1, -1, -1, 6, 23]	0.961	[10, 9, 15]	0.946	[23]
Pipe Length	**6**		**2.83**		**1.89**	

A.3 LLM Prompt Design

Here is a concrete example of a filled-out prompt for the `avila` dataset at an early stage of pipeline construction. This demonstrates how the abstract template is instantiated with real data and retrieved experiences.

LLM Prompt Example: `avila` Dataset

You are an expert data scientist specializing in automated machine learning. Your task is to analyze the provided dataset context and historical information to propose a list of 1 to 3 optimal data preparation pipelines. Each pipeline should be a sequence of data processing operators that aims to maximize the prediction accuracy of a downstream classification model.

For each proposed pipeline, you must provide:

(1) A list of operator names in sequence. (2) A confidence score (from 0.0 to 1.0) for your suggestion. (3) A brief, clear rationale explaining why this pipeline is suitable for the given data.

Current situation and context:

- Task Type: Logistic regression classification
- Key Dataset Statistics:
  - Size of dataset: 16693 rows, 10 cols
  - Missing Values: No
  - Feature Types: All numerical
  - Cols with skewed distribution: f1, f3, f4
  - Cols with outliers: f0 (9.50%), f2 (10.50%), f7 (2.62%)
- Current Partial Pipeline: [PowerTransformer, RobustScaler, ...]

Available operators: ImputerMean, StandardScaler, QuantileTransformer, PowerTransformer, PCA, MinMaxScaler, VarianceThreshold, ...

—

[Example 1]

- *Context*: A dataset with no missing values but high skewed distribution columns (B3, B9, ...), oulier columns (B1, B2, B5, ...)
- *Pipeline*: [QuantileTransformer, RandomTrees-Embedding, MinMaxScaler], accuracy 0.92

[Example 2]

- *Context*: A dataset with many correlated numerical features (A5 and A8: correlation 92%, A5 and A9: correlation 88%, ...)
- *Pipeline*: [StandardScaler, PCA], accuracy 0.88

[Example 3]

- *Context*: A dataset with no missing values, but has outliers (PC1: 154 outliers, 19.25%, PC2: 139 outliers, 17.38%, PC6: 116 outliers, 14.50%, ...)
- *Action*: PolynomialFeatures, accuracy 0.86

A.4 Hyperparameter Sensitivity Analysis

We conducted systematic sensitivity analysis for the hybrid policy weighting coefficients β_1 (LLM confidence weight), β_2 (performance deviation weight), and β_3 (entropy weight).

We selected 6 datasets spanning different characteristics: avila (high-dimensional), google (missing values), house (mixed types), micro (balanced), obesity (imbalanced), and pol (large-scale). For each dataset, we varied one parameter while keeping others fixed at their default values (0.4, 0.3, 0.3).

Table 4: Accuracy variation with hyperparameter perturbations

Parameter	-0.2	-0.1	Default	+0.1	+0.2
β_1	-2.1%	-0.8%	0.833	-0.6%	-1.9%
β_2	-1.7%	-0.5%	0.833	-0.7%	-1.6%
β_3	-1.5%	-0.4%	0.833	-0.5%	-2.3%

Key findings:

- Performance degrades gracefully with parameter changes
- Maximum performance drop of 2.3% even with ±0.2 perturbation
- β_3 (entropy weight) shows slightly higher sensitivity, suggesting the importance of uncertainty-aware guidance
- No dataset required specific tuning; default values worked uniformly

Alternative Weighting Schemes We also tested: 1. **Equal weights** (0.33, 0.33, 0.33): 1.2% average decrease 2. **Confidence-only** (1.0, 0, 0): 8.7% decrease, confirming multi-factor importance 3. **Learned weights**: Using validation set to optimize β values improved performance by only 0.3%, not justifying added complexity

Conclusion The hybrid policy demonstrates robust performance across diverse datasets without hyperparameter tuning. The default values (0.4, 0.3, 0.3) represent a good balance between LLM confidence, performance feedback, and exploration uncertainty, supporting the framework's generalizability.

A.5 Proof of Theorem 1: Optimality of First-Improvement Stopping

Theorem 1. Given LLM-suggested pipelines $P_1, ..., P_k$ ordered by confidence, the stopping rule that maximizes the value-to-cost ratio $\rho = E[\Delta]/E[N \cdot C_{eval}]$ is:

$$\tau^* = \min\{i : acc(P_i) > acc_{current}\}$$

Proof. Let us define:

- acc_i: true accuracy of pipeline P_i
- acc_{base}: current best accuracy
- $\pi_i = P(acc_i > acc_{base})$: probability that P_i improves
- $G_i = E[acc_i - acc_{base} | acc_i > acc_{base}]$: expected gain given improvement
- C_{eval}: constant cost per pipeline evaluation

Step 1: Expected Value of First-Improvement Rule

For the first-improvement stopping rule τ^*, the expected gain is:

$$\begin{aligned} E[\Delta_{\tau^*}] &= \sum_{i=1}^{k} P(\text{stop at } i) \cdot E[\text{gain at } i] && (1) \\ &= \sum_{i=1}^{k} \left(\prod_{j=1}^{i-1} (1 - \pi_j) \right) \cdot \pi_i \cdot G_i && (2) \end{aligned}$$

The expected number of evaluations is:

$$\begin{aligned} E[N_{\tau^*}] &= \sum_{i=1}^{k} P(\text{evaluate } P_i) \cdot 1 && (3) \\ &= \sum_{i=1}^{k} \prod_{j=1}^{i-1} (1 - \pi_j) && (4) \end{aligned}$$

Therefore, the value-to-cost ratio is:

$$\rho(\tau^*) = \frac{\sum_{i=1}^{k} \left(\prod_{j=1}^{i-1} (1 - \pi_j) \right) \pi_i G_i}{C_{eval} \cdot \sum_{i=1}^{k} \prod_{j=1}^{i-1} (1 - \pi_j)}$$

Step 2: Optimality Condition

Consider any alternative stopping rule τ' that continues evaluating after finding an improvement at position m. Let $S_m = \{m+1, ..., k\}$ be the set of additional evaluations.

The additional expected gain from continuing is:

$$\Delta_{extra} = \sum_{i \in S_m} \pi_i G_i \prod_{j=m+1}^{i-1} (1 - \pi_j)$$

The additional expected cost is:

$$C_{extra} = C_{eval} \cdot |S_m|$$

For continuing to be beneficial, we need:

$$\frac{\Delta_{extra}}{C_{extra}} > \frac{G_m}{C_{eval}}$$

Step 3: Confidence Ordering Implies Decreasing Marginal Value

Given that pipelines are ordered by confidence scores $c_1 \geq c_2 \geq ... \geq c_k$, and assuming the LLM's confidence correlates with expected performance (a reasonable calibration assumption), we have:

$$\pi_1 G_1 \geq \pi_2 G_2 \geq ... \geq \pi_k G_k$$

This means the expected marginal gain decreases with index:

$$\frac{\pi_i G_i}{C_{eval}} \geq \frac{\pi_{i+1} G_{i+1}}{C_{eval}} \quad \forall i$$

Step 4: First Improvement is Optimal

Given the decreasing marginal gains, once we find an improvement at position m, the expected value of continuing satisfies:

$$\frac{\sum_{i=m+1}^{k} \pi_i G_i \prod_{j=m+1}^{i-1} (1 - \pi_j)}{\sum_{i=m+1}^{k} \prod_{j=m+1}^{i-1} (1 - \pi_j)} < \pi_m G_m$$

Since we already achieved gain G_m at position m, and the expected marginal gain from continuing is less than what we've already obtained, stopping immediately maximizes the value-to-cost ratio.

Step 5: Boundary Conditions

If no improvement is found after evaluating all k pipelines, the rule naturally terminates with $\tau = k$, having explored all options.

Therefore, the first-improvement stopping rule $\tau^* = \min\{i : acc_i > acc_{base}\}$ maximizes the value-to-cost ratio ρ. □

Remark. This proof assumes that the LLM's confidence ordering is informative (i.e., higher confidence correlates with higher expected performance). Our empirical results validate this assumption, with correlation coefficient $r = 0.72$ between confidence scores and actual improvements across all experiments.