# LLM facebook\_opt\_1\_3b\_RAG

# Psudocode

Algorithm 1: Preprocess Smart Home Data

Input: Raw dataset df (pandas DataFrame)

Output: Processed dataset with engineered features

1: procedure preprocess\_data(df)

2: ▷ Drop unnecessary columns

3: df ← df.drop(['Transaction\_ID', 'Unix Timestamp'], axis=1)

4:

5: ▷ Temporal features

6: df['is\_peak\_hour'] ← df['Hour of the Day'].apply(lambda x: 1 if (6 ≤ x ≤ 9) or (18 ≤ x ≤ 21) else 0)

7: df['part\_of\_day'] ← pd.cut(df['Hour of the Day'], bins=[0,6,12,18,24], labels=['night','morning','afternoon','evening'])

8: df['is\_weekend'] ← df['Day of the Week'].apply(lambda x: 1 if x ∈ ['Saturday','Sunday'] else 0)

9:

10: ▷ Seasonal mapping

11: season\_dict ← {'December':'Winter', ..., 'November':'Fall'}

12: df['Season'] ← df['Month'].map(season\_dict)

13:

14: ▷ Cyclical encoding for hours

15: df['hour\_sin'] ← sin(2π \* df['Hour of the Day'] / 24)

16: df['hour\_cos'] ← cos(2π \* df['Hour of the Day'] / 24)

17:

18: ▷ Appliance usage features

19: appliances ← ['Television', 'Dryer', 'Oven', 'Refrigerator', 'Microwave']

20: df['total\_appliance\_usage'] ← sum(df[appliances], axis=1)

21: consumption\_threshold ← quantile(df['Energy Consumption (kWh)'], 0.75)

22: df['is\_high\_consumption'] ← df['Energy Consumption (kWh)'] > consumption\_threshold

23:

24: ▷ Normalization and efficiency ratios

25: MinMaxScaler() → Scale ['Line Voltage', 'Voltage', 'Apparent Power', 'Energy Consumption (kWh)']

26: for each appliance ∈ appliances do

27: df[f'{appliance}\_efficiency\_ratio'] ← df[appliance] / (df['Energy Consumption (kWh)'] + ε)

28:

29: ▷ Power metrics

30: df['power\_factor'] ← df['Apparent Power'] / (df['Line Voltage'] \* df['Voltage'] + ε)

31: df['energy\_per\_active\_appliance'] ← df['Energy Consumption (kWh)'] / (df[appliances].sum(axis=1) + ε)

32:

33: return df

34: end procedure

Algorithm 2: RAG-Based Energy Advisor

Input: User query query, top-k results top\_k (default=2)

Output: Personalized energy-saving recommendations

1: function rag\_energy\_advisor(query, top\_k=2)

2: ▷ Step 1: Retrieve similar cases

3: retrieved\_cases ← search\_similar\_cases(query, top\_k)

4:

5: ▷ Step 2: Build LLM prompt

6: prompt ← build\_prompt(query, retrieved\_cases)

7:

8: ▷ Step 3: Generate final response

9: response ← generate\_answer(prompt)

10: return response

11: end function

Algorithm 3: HyDE-Based Retrieval

Input: Query query, number of results top\_k

Output: Top-k reranked relevant cases

1: function search\_similar\_cases(query, top\_k)

2: ▷ Generate hypothetical answer

3: hypo\_answer ← generate\_hypothetical\_answer(query)

4: hypo\_embedding ← embedding\_model.encode(hypo\_answer)

5:

6: ▷ FAISS search

7: distances, indices ← index.search(hypo\_embedding, top\_k)

8: candidates ← data.iloc[indices]['text\_description'].tolist()

9:

10: ▷ Cross-encoder reranking

11: pairs ← [[query, doc] for doc in candidates]

12: scores ← reranker.predict(pairs)

13: reranked\_candidates ← sort(candidates by scores descending)

14: return reranked\_candidates

15: end function

Algorithm 4: Prompt Engineering

Input: Query query, retrieved cases retrieved\_cases

Output: LLM-ready prompt

1: function build\_prompt(query, retrieved\_cases)

2: context ← join(retrieved\_cases with "\n- ")

3: few\_shot\_example ← "Example: ..." (predefined template)

4: return f"""

5: You are an energy advisor. Analyze the patterns and give 3 tips:

6: {few\_shot\_example}

7: Context:

8: - {context}

9: Query: {query}

10: Answer:

11: """

12: end function

Algorithm 5: Generate Hypothetical Answer (HyDE)

Input: Query query

Output: Hypothetical answer text

1: function generate\_hypothetical\_answer(query)

2: prompt ← f"Generate hypothetical answer to: {query}\nAnswer:"

3: inputs ← tokenize(prompt)

4: outputs ← model.generate(inputs, max\_tokens=100, temperature=0.7)

5: return split(outputs.text, "Answer:")[-1].strip()

6: end function

Algorithm 6: Final Answer Generation

Input: LLM prompt prompt

Output: Generated response

1: function generate\_answer(prompt)

2: inputs ← tokenize(prompt)

3: output ← model.generate(inputs, max\_length=500, temperature=0.7)

4: return decode(output[0], skip\_special\_tokens=True)

5: end function

# Justification for each of 8 chosen chart

Retrieval Performance Over Iterations - Shows how your HyDE retrieval improved during development, demonstrating the value of iterative refinement in your RAG pipeline.

Precision-Recall Curve - Essential for evaluating your system's ability to identify high-energy consumption patterns, balancing between finding all cases (recall) and only correct ones (precision).

Energy Consumption Distribution - Reveals your data characteristics and helps explain why certain thresholds were chosen for high-consumption classification.

Appliance Usage Heatmap - Visualizes temporal usage patterns that your recommendation system needs to address, showing when interventions would be most impactful.

t-SNE Embedding Visualization - Demonstrates how your text descriptions cluster in embedding space, validating that similar usage patterns group together semantically.

Seasonal Energy Patterns - Highlights one of your key engineered features (Season) and shows why recommendations must adapt to temporal variations.

ROC Curve - Complementary to precision-recall, it shows your system's tradeoff between true positives and false positives across all thresholds.

Recommendation Quality Comparison - Clearly positions your RAG system against baselines, showcasing its superior performance across key metrics.

# 6 lines on each chart is k parallel pictures waly folders me se utha kr report me la ga lai each explanation heading k

Retrieval Performance Over Development Iterations

This line chart tracks precision and recall improvements across system versions. The upward trends demonstrate how iterative refinements (better embeddings, query formulation) enhanced retrieval quality. The converging lines suggest the system reached optimal performance without overfitting. The final precision-recall gap (0.86 vs 0.82) indicates room for improving coverage of edge cases. This validates the HyDE approach's effectiveness for energy pattern retrieval. The x-axis iterations represent cycles of embedding tuning and negative sampling.

Precision-Recall Curve

The concave shape shows good discriminative power in classifying high-consumption events. The steep initial slope indicates high confidence predictions for clear cases. The plateau at 0.7 recall suggests limitations in identifying borderline consumption patterns. The area under curve (AUC) of 0.83 reflects robust performance for an imbalanced energy dataset. The curve helps set operational thresholds - e.g., choosing 0.8 precision yields 0.65 recall. Compared to random (diagonal line), the system adds substantial value.

Energy Consumption Distribution

The right-skewed distribution reveals most readings cluster at lower consumption levels. The long tail represents critical high-usage events the system must detect. The bimodal peaks suggest distinct operational modes (normal vs intensive usage). The KDE overlay shows probability density concentrations around 0.2-0.4 kWh. This visualization guided our high-consumption threshold selection at Q3 (0.75 quantile). The shape explains why linear models would underperform on this data.

Appliance Usage Heatmap

The color gradients reveal circadian rhythms - e.g., microwave peaks at meal times. Column patterns show base loads (refrigerator) vs intermittent use (dryer). The evening hotspot (6-9PM) validates the 'peak hour' feature engineering. White spaces indicate unused appliances (oven overnight) - opportunities for savings. Row-wise variations demonstrate appliance-specific usage signatures. This chart informed our cross-appliance efficiency features.

t-SNE Embedding Visualization

The emergent clusters correlate with consumption levels (red=high). The oblong shapes suggest continuous variation in usage patterns. Sparse outliers represent rare operational scenarios. The blue-red gradient shows the embedding learned energy-intensity semantics. Cluster overlap indicates challenging borderline cases. The separation validates text descriptions effectively encode usage behavior.

Seasonal Energy Boxplot

The elevated median in summer reflects cooling loads. Winter's wider IQR indicates variable heating patterns. Fall's compact box shows stable transitional usage. Summer outliers represent heatwave responses. The plot justifies season-specific recommendation strategies. The y-axis spread confirms season is a critical temporal feature.

ROC Curve

The 0.89 AUC outperforms random (0.5) by large margin. The early steep rise indicates strong true positive rates at low false positives. The elbow at 0.1 FPR suggests an optimal operational point. The curve's stability across thresholds demonstrates robust classification. Compared to precision-recall, this better assesses overall ranking ability. The shape confirms voltage/power features are predictive.

Recommendation Quality Comparison

The progressive improvement across systems validates architectural choices. Your RAG system dominates all metrics, especially precision (15% over baselines). The explainability score (4.8/5) highlights LLM strengths. The latency-memory tradeoff shows efficient retrieval-augmentation. Novelty and diversity scores confirm broad recommendation coverage. The gap to simple LLM prompting proves retrieval's value.

# Same goes for tables

****Table 1: Recommendation Strength Metrics****  
The perfect 0.996 precision@5 across advanced systems indicates near-flawless top-5 recommendation accuracy, suggesting these systems consistently surface relevant energy-saving actions. Recall's plateau at 0.54 reveals an inherent limitation in capturing all possible interventions, likely due to irreducible scenario complexity. The maximal NDCG@5 scores demonstrate ideal ranking quality where critical recommendations consistently appear first. MAP@5's 0.99 values confirm this excellence holds across all queries, not just averages. Universal 1.0 hit rates mean every query gets at least one valid recommendation, while MRR=1.0 indicates perfect immediate relevance. These metrics collectively prove the RAG system achieves theoretical upper bounds for recommendation quality.

****Table 2: Coverage and Diversity Metrics****  
The coverage progression from 0.2 to 0.9 shows the RAG system accesses 4.5× more recommendation possibilities than random baselines. Novelty's inverse relationship with system sophistication (0.8→0.1) suggests advanced systems trade unconventional suggestions for reliability. Diversity's climb to 0.95 demonstrates the RAG system's ability to suggest complementary strategies (e.g., both appliance scheduling and voltage optimization). The explainability score's near-linear growth (2.5→4.8) reflects LLMs' superior justification capabilities. The 0.9 coverage with 0.95 diversity indicates the system avoids over-concentration on popular recommendations. These metrics reveal the RAG system's unique strength in broad, balanced recommendation generation.

****Table 3: Performance Metrics****  
BLEU scores' steady rise (0.15→0.65) mirrors improving textual recommendation quality, though absolute values suggest room for natural language refinement. Latency's linear increase (200→1200ms) reflects the computational cost of sophisticated retrieval and generation pipelines. Memory usage growth (2→8GB) exposes the hardware requirements for simultaneous embedding and language model operations. The BLEU-latency tradeoff (4.3× slower for 4.3× better text quality) reveals a characteristic Pareto frontier. Memory scaling shows approximately 1GB per system sophistication level. These metrics highlight critical engineering tradeoffs in production deployment scenarios.

****Table 4: Qualitative Metrics****  
The hallucination rate's counterintuitive increase (0.02→0.25) suggests more sophisticated systems may overextend their recommendations, requiring careful thresholding. Personalization's near-perfect 0.95 score proves the RAG system adapts to subtle usage pattern differences. Robustness's 0.4→0.9 progression demonstrates improving stability against input variations. The 0.25 hallucination rate at 0.95 personalization reveals an inherent tension between specificity and safety. These metrics uncover that the RAG system's main challenge isn't capability but recommendation calibration. The progression shows qualitative improvements beyond what quantitative metrics capture.

****Table 5: Statistical Metrics****  
MSE's dramatic drop (0.32→0.0028) shows the RAG system reduces energy prediction errors by 99.1% versus random baselines. RMSE values indicate prediction errors under 0.05kWh for advanced systems - likely within measurement noise. The F1 score's plateau at 0.67 across all non-random systems suggests this metric saturates before other quality dimensions. MSE-RMSE consistency confirms error reduction is robust across the entire distribution. The 0.0028 MSE implies the system could theoretically detect 1W standby power differences. These metrics prove the underlying energy prediction models achieve exceptional granularity.