# **1. Introdução**

Este relatório descreve a construção de um sistema de recomendação de filmes inspirado no MovieLens (100K/1M), combinando técnicas de álgebra linear com práticas clássicas de Recommender Systems. Partimos de uma matriz usuário–item (aprox. 1000×500), exploramos independência de *features*, regressão linear, filtragem colaborativa, análise de sentimentos via transformações lineares e dinâmica temporal por Cadeias de Markov, consolidando tudo em uma classe única de implementação.

# **2. Contextualização do problema**

Recomendar itens relevantes reduz sobrecarga de escolha e aumenta engajamento. Em filmes, o desafio é captar preferências latentes de usuários e características dos títulos em dados esparsos, ruidosos e não estacionários (gostos mudam com o tempo). O dataset MovieLens oferece avaliações, metadados e timestamps suficientes para testar soluções híbridas conteúdo–colaboração–tempo.

# **3. Objetivos e hipóteses**

Objetiva-se construir um pipeline reprodutível que: (i) valide *features* com ferramentas de álgebra linear; (ii) estime relações lineares entre variáveis; (iii) gere recomendações personalizadas por similaridade; (iv) extraia sinal semântico simplificado de sentimentos; (v) modele transições temporais de preferências. Hipóteses: determinantes e número de condição ajudam a evitar instabilidades; regressão por Gauss-Jordan aproxima tendências globais; vizinhança por Pearson é efetiva sob centralização; TF-IDF + SVD distingue polaridade textual simples; Markov captura mudanças de humor/estilo.

# **4. Fundamentação Teórica**

Utilizamos determinantes (Sarrus 3×3, Laplace 4×4) para detectar dependência linear; expansão de Laplace para decompor o determinante de matrizes pequenas; mínimos quadrados normais *(X⊤X)β=X⊤y(X^\top X)\beta=X^\top y*(X⊤X)β=X⊤y resolvidos por Gauss-Jordan; correlação de Pearson centrada para similaridade; TF-IDF como codificação termo–documento e TruncatedSVD (LSA) como transformação linear; Cadeias de Markov com matriz estocástica *PP*P e distribuição estacionária *π\pi*π obtida de *(P⊤−I)π=0, ∑π=1(P^\top-I)\pi=0,\ \sum\pi=1*(P⊤−I)π=0, ∑π=1.

# **5. Conceitos de álgebra linear aplicados**

Verificação de colinearidade por *∣det⁡(A)∣|\det(A)|*∣det(A)∣; análise de posto (rank) e número de condição para avaliar estabilidade; solução de sistemas lineares por eliminação de Gauss-Jordan; produtos internos, normas e correlações como operadores para similaridade; projeções lineares (SVD) reduzindo dimensionalidade; resolução de sistemas singulares com restrição de soma (Markov).

# **6. Revisão de sistemas de recomendação**

Três eixos: *content-based* (features de filmes), *collaborative filtering* (padrões de coavaliação) e temporalidade (preferências que evoluem). Integramos ainda um componente textual simples (sentimentos) para enriquecer sinais com mínimos ajustes de infraestrutura.

# **7. Metodologia**

Pipeline em etapas: preparação do dataset; normalização e auditoria de *features*; experimentos de colinearidade; regressão linear para relações globais; construção da matriz usuário–item e similaridades; classificador linear de sentimentos com TF-IDF+SVD; Cadeias de Markov por usuário a partir de gêneros binários; métricas de avaliação. Todos os blocos foram implementados em células tipo notebook com *prints* passo a passo.

# **8. Descrição detalhada de cada etapa**

Começamos criando amostras controladas (4×4) para demonstrar Sarrus e Laplace, imprimindo cada termo e menor. Em seguida, passamos aos dados reais: centralização por usuário e Pearson para vizinhança, predição por média ponderada ajustada à média do vizinho; TF-IDF dos “reviews” simulados a partir de notas, redução linear por TruncatedSVD e aprendizado de *WW*W via mínimos quadrados para pontuação; Markov por usuário: contagens *CC*C, transição *PP*P, *π\pi*π por Gauss-Jordan e checagem *πP≈π\pi P\approx\pi*πP≈π.

# **9. Justificativa das escolhas algorítmicas**

Determinantes e número de condição expõem redundâncias antes de treinos mais caros. Gauss-Jordan dá transparência pedagógica na regressão (passo a passo) e funciona bem em dimensões moderadas. Pearson centrado é robusto à escala individual. TF-IDF+SVD é linear e eficiente em alta esparsidade. Markov, ainda que simples, adiciona poder preditivo temporal com baixo custo computacional e interpretação direta.

# **10. Implementação**

Encapsulamos o fluxo na classe SistemaRecomendacaoML, com métodos para: (i) verificar independência por determinantes; (ii) resolver regressão via Gauss-Jordan; (iii) calcular similaridade usuário–usuário (tratando 0 como ausente); (iv) construir *CC*C, *PP*P, *π\pi*π e prever próximo gênero; (v) computar RMSE/MAE, Precision@K e Recall@K, além de número de condição, rank, determinante e normas.

# **11. Código comentado**

O código possui *docstrings* explicativas e *prints* educativos nas rotinas críticas (expansão de Laplace, Sarrus, Gauss-Jordan e normalização de *C→PC\rightarrow P*C→P). No notebook, cada célula imprime entradas, contas intermediárias e saídas, facilitando a rastreabilidade dos resultados e a vinculação entre teoria e prática.

# **12. Diagramas de fluxo**

O fluxo lógico segue: **Dados → Auditoria Linear (det/rank/cond) → Regressão (β) → CF (S, predições) → Texto (TF-IDF → SVD → scores) → Tempo (C→P→π, predição)**. Em termos de dependência, a matriz usuário–item alimenta CF e Markov (via gêneros), enquanto *features* numéricas alimentam auditoria e regressão; o módulo textual é paralelo e pode enriquecer ranking final.

# **13. Resultados e Discussão**

Em amostras 4×4, *∣det⁡(A)∣≠0|\det(A)|\neq 0*∣det(A)∣=0 sinalizou independência local das *features*; em *sampling* amplo, observaram-se determinantes próximos de zero em poucos casos, indicando baixa colinearidade prática. A regressão por Gauss-Jordan produziu coeficientes interpretáveis (popularidade e ano tendendo ao positivo). Na CF, as predições user-based funcionaram bem quando havia vizinhos com sobreposição suficiente. O componente de sentimentos separou positividade/negatividade em scores lineares. As cadeias de Markov mostraram transições coerentes entre gêneros e *π\pi*π estável, útil como prior temporal.

# **14. Análise dos experimentos**

**Experimento 1 (Colinearidade):** histogramas de determinantes e condição indicaram estabilidade; pares redundantes foram raros.  
 **Experimento 2 (Resolução de sistemas):** Gauss-Jordan foi claro e suficiente; Cramer/inversa foram mantidos como referência em casos pequenos, notando-se sensibilidade numérica maior.  
 **Experimento 3 (Markov):** *PP*P estocástica por construção; *π\pi*π obtida por Gauss-Jordan respeitou *∑π=1\sum\pi=1*∑π=1 e validou *πP=π\pi P=\pi*πP=π. Predições pelo último estado (argmax da linha) foram consistentes.

# **15. Interpretação dos resultados**

A boa condição das matrizes apoia o uso de modelos lineares; a regressão confirma tendências globais (ex.: mais popularidade → maior nota prevista). A CF, dependente de densidade local de avaliações, produz ganhos palpáveis quando existem vizinhos positivos. Os scores lineares de sentimento, mesmo simples, ajudam em *tie-breaks*. O componente temporal antecipa oscilações de humor/estilo, refinando recomendações de curto prazo.

# **16. Conclusões**

O pipeline integra, de forma coesa e interpretável, álgebra linear e recomendação. A soma das partes — auditoria linear, regressão transparente, vizinhança centrada, projeções textuais lineares e dinâmica Markoviana — resulta em um sistema pedagógico, eficaz e extensível, adequado para experimentação acadêmica e protótipos.

# **17. Síntese dos aprendizados**

A transparência matemática (determinantes, Gauss-Jordan, SVD, Markov) aumenta a confiança no sistema. Pré-checagens de estabilidade evitam surpresas em produção. Componentes lineares bem escolhidos entregam *baseline* forte com ótimo custo–benefício e excelente valor didático.

# **18. Trabalhos futuros**

Incorporar *implicit feedback* e *bias terms* por usuário/filme; testar *item-based CF* e *matrix factorization* (ALS/SVDpp); enriquecer texto com embeddings mais ricos; ajustar Markov para ordem superior ou modelos de estado latente; adicionar *learning-to-rank* para combinar sinais; instrumentar validação temporal e *A/B tests*; automatizar seleção de *features* via análise de colinearidade em larga escala.