Classificação Automatizada de Lançamentos Contábeis com LLM e Embeddings

A Classificação Automatizada de Lançamentos Contábeis com LLM é uma solução inovadora que utiliza Modelos de Linguagem de Grande Porte (LLMs) e embeddings para categorizar transações financeiras automaticamente, atribuindo-as às contas apropriadas em um plano de contas contábil. Este documento técnico detalha os aspectos do projeto, visando orientar a implementação de um protótipo funcional em até 3 dias. A proposta central é automatizar a categorização de lançamentos contábeis a partir de descrições em linguagem natural, combinando Processamento de Linguagem Natural (NLP) com ferramentas como LangChain, bancos de vetores e modelos pré-treinados. Espera-se que essa abordagem reduza significativamente o trabalho manual dos contadores, aumente a consistência das classificações e forneça justificativas claras para cada atribuição de conta contábil. O documento também aborda objetivos, escopo, contexto de negócio, arquitetura técnica recomendada, exemplos práticos, requisitos técnicos, critérios de avaliação da solução e possibilidades de extensão.

Importante: Não será permitido o uso de ferramentas de IA generativa (como ChatGPT ou similares) para a produção do código ou da documentação do projeto. O desenvolvimento deve ser feito manualmente.

Prazo de Entrega: O projeto deve ser entregue até quarta-feira, dia 2 de julho de 2025, às 23:59.

1. Objetivo do Projeto

O projeto visa desenvolver um sistema automatizado capaz de classificar lançamentos contábeis nas devidas contas do plano de contas, utilizando técnicas de inteligência artificial baseadas em LLMs e representações vetoriais (embeddings) de textos. Os objetivos específicos incluem:

- Automatizar a categorização de transações financeiras (e.g., identificar se uma determinada despesa corresponde a "Aluguel", "Materiais de Escritório", "Receita de Vendas", etc.).
- Reduzir a intervenção manual de profissionais de contabilidade, diminuindo erros humanos e agilizando o fechamento contábil.
- Demonstrar a viabilidade em curto prazo: a solução deverá ser construída em formato de protótipo em até 3 dias, sendo funcional e apresentável, capaz de receber descrições de lançamentos e retornar categorias sugeridas com justificativas.
- Fornecer explicações claras para cada classificação, aumentando a confiança dos usuários no sistema (ex.: indicando palavras-chave ou similaridades que levaram à escolha da conta contábil).

2. Escopo do Desafio

Incluído no Escopo:

- O projeto englobará o desenvolvimento de um modelo de classificação automatizada para lançamentos contábeis utilizando dados textuais (principalmente a descrição da transação).
- A solução incluirá a preparação de um conjunto de dados simulados de transações com categorias contábeis.
- A criação de embeddings para essas descrições.
- A implementação de um fluxo de recuperação de exemplos similares.
- O uso de um LLM para auxiliar na categorização e justificativa.
- Também está contemplado o uso de ferramentas como LangChain para orquestração do fluxo.
- E um componente de armazenamento vetorial (vector store) para busca de similaridades.
- Serão produzidos exemplos de entrada e saída esperados e realizadas avaliações básicas de desempenho (como acurácia da classificação e qualidade das justificativas).
- Desenvolvimento de uma interface de usuário (front-end) utilizando Streamlit ou similar, para permitir a interação humana com o sistema.
- Inclusão de uma coluna ou campo para a justificativa da categorização fornecida pelo LLM.
- Implementação de um status para cada categorização, que inicialmente será "Pendente" e poderá ser alterado para "Confirmado" (se o usuário aceitar a sugestão) ou "Alterado" (se o usuário selecionar outra categoria).
- Armazenamento das intervenções humanas para posterior melhoria do modelo.

Fora do Escopo:

- Integração com sistemas contábeis legados ou ERPs (o foco é a prova de conceito isolada).
- Contabilização dupla (débito/crédito) completa, limitando-se à classificação da conta destino de cada lançamento.
- Questões fiscais, conciliação bancária ou tratamento de múltiplas moedas.
- Treinamento de modelos de linguagem do zero ou fine-tuning complexo devido ao prazo curto – serão usados modelos pré-treinados existentes.
- Classificação de ativos, passivos ou cenários complexos (como depreciações), focando em um conjunto delimitado de contas contábeis (principalmente receitas e despesas operacionais).

3. Contexto de Negócio

A classificação correta de transações financeiras é crucial para a precisão das demonstrações financeiras e conformidade com normas contábeis. Atualmente, empresas usam regras fixas de associação por palavras-chave em sistemas de gestão, mas estas falham com descrições variáveis ou ruidosas. Grandes corporações possuem centenas de milhares de regras, mas ainda assim uma parte significativa dos lançamentos requer

intervenção manual. Isso ocorre pela natureza de texto livre das descrições e diversidade das transações, tornando os sistemas de regras rígidas pouco eficazes.

A relevância de uma solução automatizada reside no ganho de eficiência e redução de erros. A Amazon, por exemplo, reportou automatizar cerca de 80% do trabalho manual dos contadores ao aplicar modelos de linguagem para classificação de lançamentos, cobrindo 200 contas diferentes. Além da produtividade, a automação traz consistência e libera profissionais para tarefas de maior valor analítico. A escalabilidade é outro ponto: milhares de transações diárias exigem automação inteligente. Uma solução baseada em IA pode aprender com exemplos históricos e generalizar para novas descrições, identificando padrões complexos de codificar manualmente. O uso de LLMs permite obter justificativas textuais compreensíveis por humanos, aumentando a transparência. Em suma, o desafio aborda um ponto crítico da contabilidade empresarial: minimizar trabalhos manuais repetitivos e melhorar a qualidade da classificação contábil, impactando a confiabilidade dos relatórios financeiros.

4. Arquitetura Técnica Recomendada

A arquitetura proposta combina Recuperação de Informação baseada em embeddings com a generalização dos LLMs, integrando componentes modulares.

Componentes e Fluxo de Processamento:

- **1. Base de Conhecimento de Transações Históricas:** Um conjunto de lançamentos contábeis históricos já categorizados (ou dados simulados) serve como "conhecimento" para o modelo. As descrições dessas transações são convertidas em vetores de embedding e armazenadas em uma base vetorial junto com seus rótulos (contas contábeis) durante uma etapa de preparação offline.
- **2. Modelo de Embeddings (Encoder de Texto):** Um modelo de embedding pré-treinado (ex: BERT, Sentence Transformers, OpenAI Embeddings como `text-embedding-ada-002`) transforma descrições textuais em vetores numéricos. É importante que descrições semanticamente similares resultem em vetores próximos no espaço N-dimensional, com suporte ao idioma português. LangChain pode integrar diferentes provedores de embeddings.
- **3. Armazenamento Vetorial (Vector Store):** Os embeddings das transações históricas são armazenados em um banco de dados otimizado para consultas de vetores, permitindo busca eficiente por similaridade (cálculo de distância, ex: cosseno). Exemplos incluem FAISS, Chroma, Pinecone ou Milvus. Para o protótipo, FAISS é indicado pela simplicidade de integração local. O armazenamento indexa cada embedding de transação, vinculando-o à conta contábil correspondente.
- **4. Fluxo de Inferência (Classificação de Nova Transação):**
    - A descrição da nova transação é submetida ao mesmo modelo de embedding, gerando um vetor representativo.
    - Este vetor de consulta é enviado ao vector store, que busca as K transações mais similares na base histórica (ex: K=5 ou 10).
    - Os resultados incluem descrições e categorias contábeis das transações similares. Uma estratégia simples é aplicar votação majoritária (KNN) entre

as categorias dos vizinhos mais próximos, sugerindo a categoria mais frequente.

- **5. Integração com LLM para Justificativa e Refinamento:**
  - **Geração de Justificativas:** O LLM é acionado para produzir uma explicação em linguagem natural sobre a classificação. Fornecendo a descrição da nova transação e as descrições (e categorias) das transações similares, o LLM pode sintetizar uma justificativa clara (ex: "A transação parece relacionada a despesas de aluguel, pois contém termos como 'aluguel' e é semelhante a outros lançamentos já categorizados como Despesa Aluguel.").
  - **Classificação Assistida por Prompt:** O LLM pode atuar como classificador indireto, recebendo um prompt com a descrição da transação e uma lista do plano de contas, e sugerindo a conta adequada. Esta sugestão pode complementar a abordagem de similaridade ou ser usada em casos de baixa confiança.
  - **Refinamento de Regras ou Extração de Palavras-chave:** Uma extensão (potencialmente fora do escopo central) é extrair palavras-chave ou explicações estruturadas do LLM para criar novas regras de classificação no sistema legado.
- **6. Orquestração com LangChain:** LangChain conecta os componentes, facilitando a criação de "chains" ou pipelines (chamada ao modelo de embedding, consulta ao vector store, formatação do prompt e chamada ao LLM). LangChain provê utilitários para gerenciamento de prompts e integração com APIs de LLMs. No projeto, atua como "cola" entre embedding/vector store e LLM.

A arquitetura segue o padrão Recuperação + LLM: recupera conhecimento relevante do banco vetorial e o utiliza para classificar e explicar via LLM. Essa abordagem melhora a precisão e interpretação dos resultados, combinando memória corporativa com a capacidade de generalização do modelo de linguagem. Estudos mostram que fluxo de embedding + KNN + votação pode atingir acurácia equiparável à de especialistas humanos.

5. Fonte de Dados Simulada / Estrutura de Dados Esperada

Para viabilizar o desenvolvimento rápido, será utilizada uma fonte de dados simulada que representa lançamentos contábeis típicos. A estrutura esperada dos dados é:

- **ID da Transação:** Identificador único (numérico ou código).
- **Data:** Data do lançamento (DD/MM/AAAA ou ISO-8601), relevante para contexto mas não necessariamente usada na classificação.
- **Descrição:** Texto descritivo da transação, de extrato bancário ou comprovante. Exemplos: "Pagamento de aluguel referente a Julho/2025", "Compra de suprimentos de escritório", "Receita referente a consultoria prestada Projeto X".
- **Valor:** Valor monetário da transação (ex: R$ 5.000,00). Embora o foco seja o texto, o valor pode auxiliar em alguns casos.
- **Categoria Contábil (Conta):** A conta do plano contábil à qual a transação foi (ou deve ser) classificada. Este campo existe nos dados históricos (para treino/consulta) e é o atributo a ser previsto para novos lançamentos.
- **Justificativa LLM:** Justificativa gerada pelo LLM para a categorização proposta.
- **Status:** Status da categorização ("Pendente", "Confirmado", "Alterado").

Um conjunto sintético de exemplos será criado para cobrir cenários comuns de receitas e despesas. O plano de contas será simplificado para o protótipo, incluindo Despesas Operacionais (Aluguel, Telefonia, Viagens, Alimentação, Materiais de Escritório, Salários, etc.) e Receitas (Vendas de Produtos, Prestação de Serviços, Receitas Financeiras, etc.). Cada transação simulada será atribuída a uma dessas contas de forma coerente.

Exemplo de Estrutura de Dados (simulado - arquivo CSV ou tabela em memória):

Data	Descrição	Valor	Conta Contábil	Justificativa LLM	Status
05/07/2025	Pagamento de aluguel do escritório	5.000,0	Despesa - Aluguel	A descrição contém 'aluguel' e corresponde a um pagamento periódico de escritório, similar a lançamentos anteriores classificados como Despesa - Aluguel.	Pendente
06/07/2025	Compra de cafés e lanches (reunião equipe)	150,00	Despesa - Refeições	Identificado como despesa de refeições, pois menciona 'cafés' e 'lanches' para reunião de equipe, alinhado à categoria de Despesa - Refeições.	Pendente

07/07/2025	Boleto TIM - Conta de telefone empresa	250,00	Despesa - Telefonia	A descrição indica uma conta de telefone da empresa ('Boleto TIM'), o que se alinha a despesas com telefonia.	Pendente
10/07/2025	PIX recebido de Cliente XYZ - Projeto Alpha	20.000	Receita - Serviços Prestados	Identificado como receita de serviços, pois menciona 'recebido' e 'consultoria'. Transações semelhantes de recebimentos de clientes foram categorizadas como Receita - Serviços Prestados.	Pendente
11/07/2025	Venda de produto ABC (NF 12345)	12.000	Receita Vendas de Produtos	A transação refere-se a uma 'Venda de produto' com número de nota fiscal, o que se alinha à categoria de Receita - Vendas de Produtos.	Pendente
12/07/2025	Uber viagem reunião cliente	45,00	Despesa - Viagens	Refere-se a gasto com transporte ('Uber viagem reunião cliente'), alinhado à categoria de Despesa - Viagens. Outros lançamentos com 'Uber' ou despesas de lanche foram classificados nessa conta.	Pendente

As descrições contêm pistas (palavras-chave como "aluguel", "boleto", "recebido", "Uber") que associam a transação a determinada conta. Esses dados simulados servirão para popular a base de embeddings. Um conjunto de transações de teste, não vistas previamente, será preparado para validar as classificações sugeridas. O input para a inferência será a descrição (e possivelmente valor) de uma nova transação, e o output esperado será a conta sugerida. Isso pode ser feito via script, notebook ou uma pequena interface.

6. Exemplos de Entrada e Saída

Exemplos de como o sistema deve se comportar:

- **Exemplo 1:**
  - **Entrada:** Descrição: "Pagamento de aluguel do escritório referente a Agosto/2025"; Valor: R$ 5.000,00.
  - **Saída Esperada:** Conta sugerida: Despesa - Aluguel. Justificativa: "A descrição contém 'aluguel' e corresponde a um pagamento periódico de escritório, similar a lançamentos anteriores classificados como Despesa - Aluguel." Status: Pendente.
- **Exemplo 2:**
  - **Entrada:** Descrição: "PIX recebido de João da Silva - ref. serviços de consultoria"; Valor: R$ 3.500,00.
  - **Saída Esperada:** Conta sugerida: Receita - Serviços Prestados. Justificativa: "Identificado como receita de serviços, pois menciona 'recebido' e 'consultoria'. Transações semelhantes de recebimentos de clientes foram categorizadas como Receita - Serviços Prestados." Status: Pendente.
- **Exemplo 3:**
  - **Entrada:** Descrição: "Compra na Kalunga - materiais de escritório"; Valor: R$ 800,00.
  - **Saída Esperada:** Conta sugerida: Despesa - Materiais de Escritório. Justificativa: "A transação indica compra de materiais de escritório (fornecedor Kalunga), o que se alinha a despesas com suprimentos de escritório, categoria já conhecida." Status: Pendente.
- **Exemplo 4:**
  - **Entrada:** Descrição: "Uber Eats - Refeição equipe reunião externa"; Valor: R$ 120,00.
  - **Saída Esperada:** Conta sugerida: Despesa - Refeições/Alimentação. Justificativa: "Refere-se a gasto com alimentação ('Eats'), possivelmente entrega de refeição para a equipe, alinhado à categoria de Despesa - Refeições. Outros lançamentos com 'Uber Eats' ou despesas de lanche foram classificados nessa conta." Status: Pendente.

Em todos os casos, o sistema buscará transações similares em seu conhecimento prévio, escolherá a conta contábil adequada e explicará em linguagem natural o porquê da classificação. Os exemplos ilustram a necessidade de lidar com linguagem variada (sinônimos, nomes de fornecedores, abreviações). Um LLM bem instruído e o uso de embeddings semânticos ajudam o sistema a generalizar e classificar corretamente mesmo descrições não idênticas.

7. Requisitos Técnicos e Ferramentas Sugeridas

Para implementar a solução no prazo de 3 dias, os principais requisitos técnicos e ferramentas recomendadas são:

- **Linguagem de Programação:** Python (versão 3.8+), devido ao vasto ecossistema de NLP e ML.
- **Bibliotecas e Frameworks de IA:**
  - **LangChain:** Para orquestração de chamadas ao modelo de embedding, consultas ao vector store e chamadas ao LLM, estruturando o fluxo de forma eficiente.
  - **Modelos de Embedding:** API da OpenAI (modelo Ada Embedding) para embeddings de alta qualidade. Alternativamente, Sentence Transformers (ex.: `paraphrase-multilingual-MiniLM`) via HuggingFace para solução local/offline, com suporte ao português.
  - **Vector Store:** FAISS (via `faiss-cpu` no pip) para prototipagem local e buscas rápidas em memória. ChromaDB é outra opção simples via LangChain.
  - **LLM (Modelo de Linguagem):** Utilizar um LLM de alto desempenho para compreensão e geração de texto. Se houver acesso a API, OpenAI GPT-4 ou GPT-3.5 seriam opções viáveis (ambos suportam português e fornecem capacidade de explicar e classificar). Em cenário offline, um modelo open-source como Llama 2 ou GPT4All poderia ser utilizado, embora possivelmente com qualidade inferior nas justificativas. Como os textos de entrada e saída são curtos, o custo de usar uma API comercial seria baixo.
  - **Outras Bibliotecas:** `pandas` para manipulação de dados (ler CSV simulado), `numpy` para operações numéricas, `scikit-learn` se desejar implementar um KNN ou métricas de avaliação manualmente, `regex` para eventuais limpeza de textos, e `dotenv` para gerenciar chaves de API com segurança se necessário.
- **Ambiente de Desenvolvimento:** É recomendado usar um Jupyter Notebook ou ambiente similar (Google Colab, VSCode notebooks) para iterar rapidamente e poder documentar os passos. Isso facilita demonstração e apresentação. Alternativamente, um pequeno script Python ou aplicação simples (console/CLI) pode ser desenvolvido para receber entradas do usuário e retornar classificações.
- **Infraestrutura:** Não é necessário hardware especial além de um computador comum. Se usar modelos via API, é preciso conectividade de internet e chaves de API. Se usar um modelo local pesado (não previsto aqui devido ao prazo), aí sim seria preciso GPU adequada, mas não é o caso para embeddings e LLMs pequenos ou via API.
- **Ferramentas de Apresentação:** Uma aplicação web leve em Streamlit mostrando um formulário de input de descrição e retornando a categoria, justificativa e permitindo a alteração do status da categorização. Um dashboard mínimo mostrando algumas estatísticas de desempenho (acurácia em teste, por exemplo) e permitindo testar a classificação de exemplos ao vivo seria interessante.
- **Qualidade de Código e Organização:** Mesmo sendo um protótipo rápido, manter código organizado é importante. Sugere-se modularizar as etapas (função para gerar embedding de um texto, função para buscar similares, função para classificar com base nos vizinhos, função para montar prompt e obter justificativa do LLM). Usar comentários e logs para rastreabilidade. Se possível, incluir alguns testes unitários simples para garantir que as partes funcionam (por ex., verificar se a função de busca retorna k resultados, etc.).

Em suma, as ferramentas sugeridas orbitam em torno do ecossistema Python de NLP. O desenvolvedor (Vitor) provavelmente já possui experiência com Python, então a curva de aprendizado das bibliotecas citadas deve ser administrável dentro do prazo. A chave é evitar reinventar a roda: aproveitar embeddings pretreinados e serviços de LLM, focando a implementação na lógica de integração e não em treinar modelos do zero.

8. Critérios de Avaliação da Solução

Para determinar o sucesso do desafio e a qualidade do protótipo desenvolvido, serão considerados os seguintes critérios de avaliação:

- **Acurácia da Classificação:** Mede a porcentagem de lançamentos corretamente classificados na conta contábil correta. Embora não se espere perfeição em um protótipo, um bom resultado seria uma acurácia significativamente melhor que um baseline aleatório ou regras simples. Idealmente, compara-se a saída do modelo com categorias esperadas em um conjunto de teste. Pesquisas indicam que abordagens baseadas em embeddings + LLM podem alcançar acurácia próxima à de humanos em tarefas semelhantes, então um alvo de, por exemplo, >80% de acerto em dados simulados seria desejável.
- **Clareza das Justificativas:** Verifica se para cada classificação o sistema fornece uma explicação compreensível e consistente. As justificativas devem referenciar elementos da descrição ou exemplos similares que sustentem a decisão. Um avaliador humano deve considerar a justificativa plausível e útil. Aqui, utiliza-se feedback qualitativo: as explicações estão ajudando a confiar na classificação? Estão evitando linguagem vaga ou genérica demais? A transparência é um diferencial importante (modelos interpretáveis são citados como vantajosos em contextos contábeis).
- **Robustez da Solução:** Envolve testar o sistema em situações variadas para ver se mantém desempenho estável. Exemplos de robustez: habilidade de lidar com descrições não vistas (generalização); tolerância a pequenos erros de ortografia ou uso de termos incomuns; tempo de resposta adequado (eficiência); e comportamento consistente ao adicionar novos dados (se adicionarmos mais transações à base, o modelo continua performando bem?). Também avalia-se se o sistema falha graciosamente - por exemplo, se recebe uma entrada completamente fora do domínio, ele não deve quebrar, mas talvez retornar "Não categorizado" ou um pedido de mais informações.
- **Compleção do Escopo / Funcionalidade:** O protótipo atende todos os pontos propostos? Isto é, consegue ler entradas, processar e dar saída formatada com categoria e justificativa, usando de fato embeddings e LLM no pipeline. Avalia-se se todos os componentes descritos na arquitetura foram implementados (mesmo que de forma simples). Se alguma parte não foi possível dado o tempo, isso deve ser justificado e, de preferência, uma solução alternativa ou simulação da funcionalidade deve ser apresentada.
- **Facilidade de Apresentação e Uso:** Como este desafio implica mostrar resultados rapidamente, conta pontos a solução ser fácil de demonstrar. Isso inclui ter exemplos preparados, possivelmente uma interface interativa ou pelo menos um notebook bem estruturado onde se possa alterar a entrada e ver a saída imediatamente. Se o avaliador (ou outros stakeholders) conseguirem testar com

suas próprias descrições e obtiver respostas coerentes, melhor. Portanto, considera-se se a solução é usuário-amigável no contexto de uma demo.

- **Documentação do Desenvolvimento:** Ainda que fora do produto final, espera-se que o desenvolvedor documente as decisões técnicas, parâmetros escolhidos e como rodar o protótipo. Parte disso está atendido pelo presente documento técnico. Complementarmente, comentários no código ou um README no repositório (se houver) contribuem para a avaliação positiva.
- **Capacidade de Intervenção Humana e Feedback:** Verificação se a interface permite a intervenção humana na categorização (confirmação ou alteração) e se essas intervenções são armazenadas para futura melhoria do modelo.

Cada critério acima poderá ter um peso específico na avaliação global. Por exemplo, acurácia e robustez podem ter peso maior, mas uma justificativa bem feita e a cobertura do escopo também são fundamentais. Em última análise, o objetivo é demonstrar um equilíbrio: um sistema que funcione corretamente, que explique suas ações e que tenha sido construído seguindo boas práticas dentro do contexto de tempo limitado.

9. Extensões e Melhorias Possíveis

Caso haja tempo extra ou para apontar caminhos de evolução além do protótipo inicial, consideram-se as seguintes extensões e melhorias da solução:

- **Treinamento de Embeddings Especializados:** No protótipo usamos um modelo genérico de embeddings. Poderíamos incrementar a qualidade das recomendações treinando um encoder específico no domínio financeiro/contábil. Técnicas de *metric learning* (aprendizado por similaridade) permitiriam ajustar o espaço vetorial para que transações da mesma categoria fiquem mais próximas entre si, enquanto transações de categorias diferentes fiquem distantes. Por exemplo, usando um conjunto grande de dados categorizados, aplicaria-se *triplet loss* ou similar para produzir embeddings altamente ajustados às necessidades de classificação do plano de contas. Isso possivelmente aumentaria a acurácia além do que modelos genéricos fornecem.
- **Expansão do Plano de Contas e Hierarquia:** No futuro, o sistema poderia lidar com um plano de contas completo, incluindo contas patrimoniais (ativos, passivos) e hierarquias de contas. Uma abordagem seria primeiro classificar em nível macro (e.g., distinguir se a transação é Receita, Despesa, Ativo etc.) para depois refinar para a conta específica. LLMs podem ajudar a entender contexto que indique, por exemplo, que "compra de computador" poderia ser um ativo imobilizado ao invés de despesa, dependendo da política da empresa.
- **Integração com Sistemas Legados:** Transformar o protótipo em uma ferramenta integrada ao fluxo real contábil. Isso envolve conectar a solução a um ERP ou sistema de gerenciamento financeiro, onde os lançamentos são ingestados automaticamente e as sugestões de contas são apresentadas ao contador diretamente na interface que ele já usa. Para tal, seria preciso desenvolver APIs ou mecanismos de exportação/importação de dados. Essa integração também abriria espaço para um componente de *feedback loop*: se o contador corrigir a sugestão, esse novo dado pode retroalimentar a base de treinamento (aprendizado contínuo).
- **Suporte Multilíngue e Regional:** Embora no Brasil a língua seja o português para descrições, empresas multinacionais podem ter descrições em inglês ou outro

idioma. Adaptar o pipeline para ser multilíngue seria uma extensão. Usar embeddings multilíngues ou detecção de idioma + modelo específico para cada, por exemplo. Além disso, considerações regionais como formatos diferentes (por ex, número de documento fiscal) poderiam ser incorporadas ao pré-processamento.

- **Incorporação de Dados Adicionais:** A classificação poderia ser enriquecida usando não apenas a descrição e valor, mas outras features possivelmente disponíveis, como o nome do fornecedor/cliente, a categoria de gasto pré-classificada em cartão (MCC code, no caso de cartões de crédito), ou até informações do comprovante fiscal. Essas informações estruturadas poderiam entrar como entradas adicionais para o modelo (concatenadas na descrição ou em um vetor separado). Um modelo multimodal ou um LLM com instruções adequadas poderia então considerar esses detalhes para maior precisão. Por exemplo, sabendo que "Loja X" pertence ao ramo de material de construção poderia ajudar a classificar a compra como um ativo ou despesa específica.
- **Métricas e Monitoramento Contínuo:** Em uma versão produtiva, seria importante incluir monitoramento de desempenho ao longo do tempo. Por exemplo, acompanhar a acurácia mensal, ou quantas sugestões o sistema forneceu vs. quantas foram alteradas pelo usuário (taxa de aceitação). Implementar métricas de confiança nos outputs do modelo (como uma probabilidade ou score de similaridade médio dos vizinhos) para possivelmente só automatizar totalmente quando a confiança for alta, deixando casos de baixa confiança para revisão manual prioritária.
- **Uso de LLM para Explicações Avançadas:** Elevar o nível das justificativas fornecidas pelo modelo, fazendo com que o LLM também traga referências normativas ou políticas da empresa na explicação. Ex: "Esta despesa foi classificada como Viagens conforme política interna, que define Uber e transporte como despesas de viagem." Isso porém exigiria alimentar o LLM com conhecimento adicional (documentos de políticas ou normas contábeis), possivelmente via técnica de Retrieval-Augmented Generation mais abrangente.

Em conclusão, há diversas frentes de melhoria pós-protótipo. Entretanto, dentro do prazo de 3 dias, o foco recai em entregar a solução básica funcional: classificação acurada de lançamentos contábeis usando LLM + embeddings, com outputs explicativos. As extensões listadas servem como um roadmap de como tornar a solução mais completa e robusta no futuro. Com uma base bem construída agora, essas evoluções se tornam mais factíveis por exemplo, o uso de uma arquitetura modular com LangChain deve facilitar adicionar novos passos ou fontes de dados. Vale destacar que mesmo a solução básica já demonstrará um avanço significativo em relação aos métodos manuais ou baseados apenas em regras fixas, confirmando a proposta de valor da abordagem de LLM na contabilidade.