🤖 Pipeline ETL Auto-Corretivo com IA

Trabalho de Conclusão de Curso - Sistemas de Informação

Aluna: Thalissa de Melo Erran

Curso: Sistemas de Informação

Área: Engenharia de Dados

Prazo: 3 meses

🎯 Escopo do Projeto

Objetivo Principal

Desenvolver um **pipeline ETL inteligente** que automaticamente detecta, categoriza e corrige inconsistências em dados, utilizando técnicas de Machine Learning para identificar padrões dominantes e anomalias.

🔍 Funcionalidades Específicas

🔧 Auto-Correção Inteligente

Detecta padrões dominantes (≥75%) e corrige automaticamente inconsistências menores. Ex: 80% virgula, 20% ponto → corrige os 20%

🚨 Detecção de Anomalias

Identifica valores discrepantes da moda/padrão estatístico e categoriza por tipo de anomalia, sem alterar os dados

📧 Relatórios Automáticos

Envia emails detalhados com todas as correções aplicadas, anomalias detectadas e estatísticas do processamento

📊 Dashboard Interativo

Interface web para visualizar dados processados, correções aplicadas e anomalias detectadas

⚙️ Regras de Auto-Correção Detalhadas

🎯 Thresholds de Decisão

Percentual Dominante	Ação	Confiança	Exemplo
≥ 80%	✅ **Corrige Automaticamente**	Alta	80% virgula, 20% ponto
70-79%	✅ **Corrige + Aviso Extra**	Média-Alta	75% maiúscula, 25% minúscula
60-69%	⚠️ **Apenas Reporta**	Média	65% formato A, 35% formato B
50-59%	⚠️ **Apenas Reporta**	Baixa	55% virgula, 45% ponto
< 50%	⚠️ **Apenas Reporta**	Muito Baixa	Múltiplos padrões

🎛️ Regras Específicas por Tipo de Dados

Tipo de Dado	Threshold Auto-Correção	Justificativa
Formatação Decimal	≥ 75%	Impacto baixo, facilmente reversível
Formatação Data	≥ 80%	Impacto médio, pode afetar cálculos
Case (Maiúsc/Minúsc)	≥ 70%	Impacto baixo, questão estética
Encoding/Acentos	≥ 80%	Impacto alto, pode afetar buscas
Formatos Telefone/CPF	≥ 85%	Impacto alto, dados sensíveis

🧮 Algoritmo de Decisão

```
def decide_action(pattern, data_type, column_name): total_records = sum(pattern.values()) dominant_pattern = max(pattern, key=pattern.get) dominant_percentage = (pattern[dominant_pattern] / total_records) * 100 # Regras por tipo de dados thresholds = { 'decimal': 75, 'date': 80, 'case': 70, 'encoding': 80, 'phone_cpf': 85 } threshold = thresholds.get(data_type, 75) # Default 75% if dominant_percentage >= threshold: return { 'action': 'AUTO_CORRECT', 'confidence': 'HIGH' if dominant_percentage >= 85 else 'MEDIUM', 'pattern': dominant_pattern, 'percentage': dominant_percentage } else: return { 'action': 'REPORT_ONLY', 'confidence': 'LOW', 'reason': 'Padrão não suficientemente dominante', 'percentage': dominant_percentage }
```

📧 Exemplos de Relatórios Diferenciados

✅ Email para Auto-Corrigidos:

CORREÇÕES APLICADAS (Confiança Alta):

- 347 virgulas → pontos (85% dominante, coluna: preco)
- 156 nomes → maiúscula (78% dominante, coluna: nome)

⚠️ CORREÇÕES APLICADAS (Confiança Média):

- 89 datas → DD/MM/YYYY (72% dominante, coluna: data_nasc)
- → Recomendamos validação manual

🚨 Email para Apenas Reporte:

INCONSISTÊNCIAS DETECTADAS (Requer Intervenção):

- Formatação decimal inconsistente (55% virgula, 45% ponto)
- Múltiplos formatos de data (40% DD/MM, 35% MM/DD, 25% YYYY-MM-DD)
- Case inconsistente (52% maiúsc, 48% minúsc)

🏷️ Categorização Final de Resultados

Categoria	Critério	Ação	Notificação
Auto-Corrigido	≥ threshold definido	Correção automática	Email com detalhes
Reportado - Baixa Confiança	< threshold definido	Apenas reporta	Email de alerta
Reportado - Múltiplos Padrões	>2 padrões significativos	Apenas reporta	Email detalhado
Reportado - Padrão Equilibrado	Diferença <20%	Apenas reporta	Email de revisão
Anomalias Estatísticas	Outliers detectados	Apenas reporta	Email de anomalias

🛠️ Stack Técnica Utilizada

🔄 n8n

Função: Orquestração de workflows ETL

Vantagem: Interface visual, automação

🤖 scikit-learn

Função: Machine Learning para detecção

Vantagem: Algoritmos prontos, eficiente

🐼 pandas

Função: Manipulação de dados

Vantagem: Poderoso para ETL

🌐 Streamlit

Função: Dashboard web interativo

Vantagem: Rápido desenvolvimento

🗄️ PostgreSQL

Função: Armazenamento de dados

Vantagem: Robusto e gratuito

📈 numpy

Função: Cálculos estatísticos

Vantagem: Performance matemática

🏗️ Arquitetura do Sistema

Fluxo de Processamento

📊 Extract → 👁 Analyze → 🏷️ Categorize → 🔧 Auto-Fix → 🚨 Detect → 📧 Report → 💾 Load

📝 Detalhamento dos Processos

Etapa	Descrição	Tecnologia	Output
Extract	Coleta dados de múltiplas fontes	n8n + Python	Dados brutos
Analyze	Analisa padrões e distribuições	scikit-learn	Padrões identificados
Categorize	Classifica problemas por tipo	pandas + ML	Categorias definidas
Auto-Fix	Corrige automaticamente padrões minoritários	pandas	Dados corrigidos
Detect	Identifica anomalias estatísticas	scikit-learn	Lista de anomalias
Report	Gera relatórios automáticos	n8n + Email	Relatório enviado
Load	Armazena dados processados	PostgreSQL	Dados disponíveis

🚀 Vantagens e Benefícios

💼 Valor Comercial

- **Automação Real:** Reduz 90% do trabalho manual de limpeza de dados
- **Escalabilidade:** Processa milhares de registros automaticamente
- **Transparência:** Relatórios detalhados de todas as alterações
- **Confiabilidade:** Não altera dados críticos, apenas reporta anomalias
- **Configurável:** Thresholds ajustáveis por tipo de dados

🎓 Valor Acadêmico

- **Multidisciplinar:** Combina ETL, ML, desenvolvimento web e banco de dados
- **Inovador:** Poucos trabalhos combinam auto-correção com categorização inteligente
- **Prático:** Resolve problema real presente em todas as empresas
- **Demonstrável:** Interface visual para apresentação clara
- **Fundamentado:** Regras claras e algoritmos bem definidos

📅 Cronograma Detalhado (3 meses)

Mês 1 - Fundação

Semana 1-2: Aprendizado e Setup

- 📚 Estudo do n8n (tutoriais, documentação)
- 🐼 Revisão Python + pandas
- 🔧 Configuração do ambiente local
- 🗄️ Setup PostgreSQL

Semana 3-4: Pipeline Básico

- ✅ Primeiro workflow n8n funcionando
- 📊 ETL básico com dados CSV
- 🔗 Conexão com banco de dados
- ✅ Primeira versão do pipeline

Mês 2 - Inteligência

Semana 1-2: Machine Learning

- 🤖 Estudo scikit-learn (algoritmos de detecção)
- 📈 Implementação de detecção de padrões
- 🤖 Implementação das regras de threshold
- 🔧 Sistema de auto-correção
- ✅ Testes com dados simulados

Semana 3-4: Detecção de Anomalias

- 🚨 Algoritmos de detecção de outliers
- 🏷️ Sistema de categorização
- 📧 Implementação de relatórios automáticos diferenciados
- 📈 Otimização de performance

Mês 3 - Finalização

Semana 1-2: Dashboard e Interface

- 🌐 Desenvolvimento dashboard Streamlit
- 📊 Visualizações interativas
- ⚙️ Configurações de threshold personalizáveis
- 🔧 Filtros e funcionalidades avançadas
- 🎨 Interface amigável

Semana 3-4: Entrega

- ✅ Testes finais com diferentes cenários
- 📝 Documentação completa das regras
- 🎤 Preparação da apresentação
- 🎯 Ensaio para a banca

⚠️ Considerações de Aprendizado

Estratégia para Dominar as Ferramentas

Ferramenta	Dificuldade	Tempo Aprendizado	Recursos
n8n	Fácil	1-2 semanas	Documentação oficial, YouTube
scikit-learn	Médio	2-3 semanas	Tutoriais, exemplos práticos
Streamlit	Fácil	1 semana	Documentação, exemplos
pandas	Básico	Revisão	Já conhece
PostgreSQL	Básico	Revisão	Já conhece

💰 Orçamento do Projeto

💸 Projeto 100% Gratuito

- ✅ Todas as ferramentas são open source
- ✅ Desenvolvimento local (sem custos de cloud)
- ✅ Dados simulados para testes
- ✅ Hospedagem gratuita para demonstração

🎯 Justificativa da Escolha

Por que este projeto é ideal para o TCC?

- **Escopo Adequado:** Nem muito simples, nem muito complexo para 3 meses
- **Relevância Profissional:** Conecta com sua área de Engenharia de Dados
- **Demonstração Visual:** Interface clara para apresentação à banca
- **Aplicabilidade:** Solve um problema real enfrentado por empresas
- **Conhecimento Técnico:** Demonstra domínio em múltiplas tecnologias
- **Inovação:** Combina ETL tradicional com IA moderna
- **Regras Claras:** Sistema bem fundamentado e configurável

📋 Próximos Passos

Para Começar o Projeto:

1. **Validar proposta** com orientador
2. **Configurar ambiente** de desenvolvimento
3. **Criar cronograma** detalhado semanal
4. **Iniciar aprendizado** das ferramentas
5. **Preparar dados** simulados para testes
6. **Implementar regras** de threshold definidas

Este projeto representa uma oportunidade única de desenvolver uma solução inovadora que combina conhecimentos acadêmicos com aplicação prática, demonstrando competências técnicas em múltiplas áreas da computação com regras claras e bem fundamentadas.