Projeto de Pesquisa

Trabalho de conclusao de curso de especializacao em Data Science

Alceu Eilert Nascimento

Título do Projeto

Análise de Decisões de Admissibilidade de REsp pela 1ª Vice-Presidência do TJPR utilizando Técnicas de Ciência de Dados

Objetivo Geral

Desenvolver um banco de dados consolidado das decisões de admissibilidade de Recursos Especiais (REsp) proferidas pela 1ª Vice-Presidência do TJPR e realizar análises para identificar quais características influenciam no resultado dos recursos.

Objetivos Específicos

- **Coleta de Dados**: Obter aproximadamente 148 mil decisões judiciais diretamente do site do TJPR.
- **Processamento de Dados**: Extrair, limpar e organizar os dados, transformando cada decisão em um arquivo JSON estruturado.
- **Criação de Banco de Dados**: Consolidar os dados em um banco de dados DuckDB eficiente para consultas analíticas.
- **Disponibilização dos Dados**: Desenvolver um website para compartilhar o banco de dados com outros pesquisadores, incluindo documentação completa.
- **Análise de Dados**: Utilizar algoritmos de classificação, especificamente o algoritmo "a priori" (Agrawal and Srikant 1994), e análises de grafos paralelos para identificar features que influenciam nos resultados dos recursos.

1. Perfil dos Dados e Dificuldades de Obtenção

1.1. Descrição dos Dados

- **Origem**: Decisões de admissibilidade de Recursos Especiais da 1ª Vice-Presidência do TJPR, disponível em `https://portal.tjpr.jus.br/jurisprudencia/`.
- **Quantidade**: Aproximadamente 148 mil decisões.
- **Formato Original**: Páginas HTML disponíveis no site oficial do TJPR.
- **Conteúdo**: Texto semi-estruturado (HTML) contendo informações sobre o recurso, partes envolvidas, fundamentação jurídica e decisão final (admitido ou não admitido).

1.2. Dificuldades de Obtenção

- **Acesso Não Estruturado**: As decisões não estão disponíveis para download em massa ou em formatos estruturados (e.g., CSV, JSON).
- **Limitações Técnicas**: Restrições de acesso, como limites de requisições por IP ou mecanismos de proteção contra bots.
- **Inconsistências nos Dados**: Variações no formato das páginas HTML ao longo do tempo, dificultando a extração padronizada.

2. Desenvolvimento do Webcrawler

2.1. Necessidade do Webcrawler

- **Automação da Coleta**: Devido ao volume de dados e à falta de APIs ou mecanismos oficiais de exportação, um webcrawler é essencial para automatizar o processo de coleta.

2.2. Ferramentas Utilizadas

- **Linguagem de Programação**: Python.
- **Bibliotecas**:
  - **Selenium**: Para simular a interação com o navegador e superar possíveis mecanismos anti-bot.
  - **BeautifulSoup**: Para parsing e extração de dados das páginas HTML.

2.3. Procedimento de Acesso e Download

- **Estratégia de Navegação**:
  - Mapear URLs das decisões.
  - Utilizar Selenium para navegar e carregar conteúdo dinâmico se necessário.
- **Execução Faseada**:
  - Dividir o processo em lotes para melhor gerenciamento e monitoramento.
  - Implementar intervalos entre requisições para evitar sobrecarregar o servidor e ser bloqueado.
- **Controle de Erros**:
  - Capturar exceções e erros de conexão.
  - Implementar mecanismos de retry com backoff exponencial.
  - Log de erros detalhado para posterior análise.
- **Evitar Retrabalho**:
  - Manter um registro (e.g., um arquivo CSV ou uma tabela no banco de dados) das decisões já baixadas.
  - Verificar antes de cada download se a decisão já foi obtida.

2.4. Salvamento das Decisões

- **Armazenamento**:
  - Organizar os arquivos HTML em diretórios estruturados (e.g., por ano ou número do processo).
  - Nomear os arquivos de forma consistente, usando identificadores únicos.

3. Extração e Transformação dos Dados

3.1. Extração dos Dados

- **Parsing com BeautifulSoup**:
  - Identificar os padrões nas páginas HTML para localizar elementos de interesse (e.g., número do processo, data, partes, texto da decisão).
  - Tratar casos especiais e variações no layout.
- **Criação de Esquema JSON**:
  - Definir uma estrutura JSON padronizada para todas as decisões.

  - **Campos sugeridos**:
    * `numero_processo`
    * `data_decisao`
    * `partes`
    * `relator`
    * `texto_decisao`
    * `resultado` (admitido/não admitido)
    * `fundamentacao`

3.2. Transformação e Limpeza dos Dados

- **Normalização**:
  - Padronizar formatos de data.
  - Converter textos para caixa baixa/alta conforme necessário.
- **Remoção de Ruídos**:
  - Eliminar tags HTML residuais, espaços em branco excessivos e caracteres especiais.
- **Tratamento de Valores Faltantes**:
  - Identificar campos faltantes e decidir sobre estratégias de imputação ou exclusão.
- **Validação**:
  - Verificar consistência dos dados (e.g., datas válidas, campos obrigatórios preenchidos).

4. Organização e Consolidação dos Dados

4.1. Criação do Banco de Dados DuckDB

- **Justificativa**:
  - DuckDB é um sistema de gerenciamento de banco de dados analítico embutido, ideal para grandes volumes de dados e consultas analíticas complexas.
- **Importação dos Dados**:
  - Carregar os arquivos JSON diretamente para o DuckDB.
  - Utilizar scripts Python ou SQL para automatizar o processo.
- **Estruturação do Banco**:
  - Definir tabelas e relacionamentos para otimizar consultas.

- **Indexação e Otimização**:
  - Criar índices nos campos mais consultados (e.g., `data_decisao`, `resultado`).
- **Backup e Segurança**:
  - Implementar rotinas de backup.
  - Garantir a segurança dos dados, especialmente se houver informações sensíveis.

5. Desenvolvimento do Website para Disponibilização dos Dados

5.1. Objetivo do Website

- **Compartilhamento**: Disponibilizar o banco de dados para outros pesquisadores.
- **Documentação**: Fornecer informações detalhadas sobre o dataset e como utilizá-lo.

5.2. Tecnologias Utilizadas

- **Back-end**: Framework Python.
- **Front-end**: HTML5 e CSS3.
- **Hospedagem**: Serviços como Heroku, AWS, Gitpges.

5.3. Funcionalidades do Website

- **Download de Dados**: Possibilidade de baixar o dataset completo ou filtrado.
- **Documentação**:
  - Guia do usuário.
  - Descrição detalhada de cada variável (dicionário de dados).
  - Exemplos de uso.
- **Contato e Suporte**: Formulário ou e-mail para contato em caso de dúvidas.

6. Análise de Dados

6.1. Análise de Features Relevantes

- **Análise Exploratória de Dados (EDA)**:
  - Estatísticas descritivas das variáveis.
  - Identificação de outliers e padrões.

- **Seleção de Features**:
  - Utilizar técnicas como correlação, análise de variância (ANOVA) ou testes de qui-quadrado para identificar features que influenciam no resultado.

6.2. Classificação com Algoritmo Apriori

- **Objetivo**: Descobrir regras de associação entre features e o resultado do recurso.
- **Implementação**:
  - Utilizar bibliotecas Python como `mlxtend`[1] de (Raschka 2018) para implementar o algoritmo Apriori.
  - Definir suporte, confiança e lift para filtrar as regras mais relevantes.
- **Análise dos Resultados**:
  - Interpretar as regras encontradas.
  - Avaliar a significância e aplicabilidade jurídica.

6.3. Análise de Grafos Paralelos

- **Finalidade**: Visualizar múltiplas dimensões simultaneamente e identificar padrões.
- **Ferramentas**:
  - Bibliotecas como Plotly, Matplotlib ou Seaborn.
- **Procedimento**:
  - Selecionar as features mais relevantes.
  - Plotar grafos paralelos para observar a relação entre elas e o resultado do recurso.
- **Interpretação**:
  - Identificar agrupamentos e tendências.
  - Relacionar os achados com fundamentos jurídicos.

7. Procedimentos Metodológicos

7.1. Planejamento

- **Cronograma**:

[1] https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/

  - Estabelecer prazos para cada etapa: coleta, processamento, análise e desenvolvimento do website.

- **Recursos Necessários**:
  - **Hardware**: Computador com capacidade de processamento e armazenamento adequados.
  - **Software**: Python, bibliotecas mencionadas, ferramentas de desenvolvimento web.
  - **Orientação**: Acompanhamento por professores ou profissionais especializados.

7.2. Considerações Éticas e Legais

- **Conformidade Legal**:
  - Verificar a legalidade de coletar e compartilhar as decisões judiciais.
  - Respeitar a Lei Geral de Proteção de Dados (LGPD) quanto a informações pessoais.
- **Anonimização**:
  - Remover ou anonimizar dados pessoais sensíveis presentes nas decisões.
- **Licenciamento**:
  - Definir uma licença para o uso dos dados compartilhados (e.g., Creative Commons).

8. Resultados Esperados

- **Banco de Dados Consolidado**: Um dataset estruturado e limpo das decisões de admissibilidade de REsp do TJPR.
- **Ferramenta download**: Um website funcional que permita o acesso à base de dados que contém as decisões.
- **Informações**: Identificação de features que influenciam nos resultados dos recursos, contribuindo para a compreensão do processo decisório.
- **Contribuição Acadêmica**: Disponibilização de um recurso valioso para pesquisas futuras em direito e ciência de dados.

9. Conclusão

Este projeto integrará técnicas avançadas de ciência de dados com o estudo de decisões judiciais, proporcionando insights significativos sobre o funcionamento do sistema judiciário e auxiliando na promoção da transparência e eficiência.

10. Referências

- **Bibliográficas**:
  - Literatura sobre mineração de textos jurídicos.
  - Estudos anteriores que utilizaram algoritmos de associação em dados jurídicos.
- **Tecnológicas**:
  - Documentação oficial do Selenium, BeautifulSoup, DuckDB.
  - Tutoriais e exemplos de implementação do algoritmo Apriori em Python.

Agrawal, Rakesh, and Ramakrishnan Srikant. 1994. "VLDB ´94." In, 487–99. Santiago.

Raschka, Sebastian. 2018. "MLxtend: Providing Machine Learning and Data Science Utilities and Extensions to Python's Scientific Computing Stack." *Journal of Open Source Software* 3 (24): 638. https://doi.org/10.21105/joss.00638.