Prova — Aprendizado de Máquina (em duplas)

Desenvolver, implementar e avaliar algoritmos de classificação para Doença Renal Crônica (DRC/CKD) usando técnicas de Case-Based Reasoning (CBR) e métodos de otimização de pesos da função de similaridade. Devem ser construídos modelos para os rótulos do dataset: CKD_Stage (classificação multiclasse) e CKD_Progression (classificação binária). O conjunto de dados com registros clínicos e laboratoriais será fornecido em anexo.

Conjunto de dados — features

O dataset contém (entre outras) as seguintes variáveis — implemente as transformações e o pré-processamento necessários antes da modelagem:

Sex — Gênero do paciente (masculino / feminino)

Age — Idade (anos)

Systolic_Pressure — Pressão arterial sistólica (mm Hg)

BMI — Índice de Massa Corporal

CKD_Cause — Causa da DRC

Hemoglobin — Hemoglobina (g/dL)

Albumin — Albumina sérica (g/dL)

Creatinine — Creatinina sérica (mg/dL)

eGFR — Taxa de filtração glomerular (mL/min/1,73m²)

CKD_Stage — Estágio atual da DRC (rótulo multiclasse)

CKD_Risk — Categoria de risco da DRC

Dipstick_Proteinuria — Resultado do dipstick para proteinúria

Proteinuria — Proteinúria (sim / não)

Occult_Blood_in_Urine — Sangue oculto na urina

Protein_Creatinine_Ratio — Relação proteína/creatinina

UPCR_Severity — Gravidade da UPCR

Hypertension — Hipertensão (sim / não)

Previous_CVD — Histórico de doença cardiovascular (sim / não)

Diabetes — Diabetes (sim / não)

RAAS_Inhibitor — Uso de inibidores da RAA

Calcium_Channel_Blocker — Uso de bloqueadores de cálcio

Diuretics — Uso de diuréticos

CKD_Progression — Progressão da DRC (sim / não) — rótulo binário

Observação: o dataset foi pré-tratado, mas vocês devem executar as etapas de pré-processamento necessárias (por exemplo: remover variáveis que apresentem correlação maior que 90% com o rótulo, tratar missing, normalizar, etc.).

Tarefas propostas (exemplos)

1. Análise exploratória de dados (EDA)

Estatísticas descritivas (média, desvio padrão, percentis), análise de valores faltantes.

Visualizações: histogramas, boxplots, scatter plots, mapa de correlação, etc., para entender distribuições e relações.

2. Pré-processamento

Tratar valores faltantes; normalizar/ padronizar variáveis quando apropriado.

Remoção ou engenharia de features conforme necessidade.

Dividir o dataset em treino e teste; use o treino para ajustar pesos da similaridade e o teste para avaliação final.

3. Implementação dos modelos CBR

Criar função de similaridade capaz de lidar com atributos numéricos, categóricos e textuais.

Implementar o algoritmo CBR base: recuperar casos mais similares e decidir rótulos por votação (majority voting), iniciando com pesos iguais (por exemplo, w = 1).

4. Otimização dos pesos da função de similaridade

Implementar um método de otimização para ajustar os pesos (escolher um dos três listados abaixo) e justificar a escolha.

- Busca em grade (Grid Search)
- Gradiente descendente (otimização contínua)
- Algoritmo genético (busca evolucionária)

Projetar experimentos controlados para comparar performance com/sem ajuste de pesos e testar variações de parâmetros (iterações, população, thresholds de similaridade, número máximo de casos recuperados etc.).

5. Avaliação dos modelos

Métricas: Acurácia, Precisão, Revocação (Recall), F1-Score, Matriz de Confusão e AUC-ROC (quando aplicável).

Análise qualitativa: discutir falsos positivos e falsos negativos e suas implicações clínicas.

Comparar desempenho entre abordagens (CBR base vs. CBR com pesos otimizados) e entre problemas (CKD_Stage vs CKD_Progression).

6. Discussão e conclusão

Comparar resultados, justificar por que uma técnica se saiu melhor ou pior.

Apontar limitações do estudo e propor melhorias e trabalhos futuros.

Função de similaridade

A função deve permitir pesos diferentes por atributo e suportar tipos variados de features (numéricos, categóricos, textuais).

Sugestões de tratamento: distância Euclidiana ou normalizada para numéricos; métricas de similaridade de string para textuais; abordagens apropriadas para categóricos (match/mismatch, distância Jaccard, etc.).

Documentem a lógica e as decisões adotadas na construção da função de similaridade.

Algoritmo CBR

Implementar um CBR básico com pesos iguais (baseline).

Implementar a etapa de recuperação de casos (rank por similaridade) e a decisão por votação majoritária para classificação.

Avaliar esse baseline no conjunto de teste e registrar as métricas.

Algoritmo(s) de otimização de pesos

Escolham e implementem pelo menos um dos métodos propostos (Grid Search, Gradiente Descendente ou Algoritmo Genético).

Justifiquem a escolha do método e descrevam parâmetros escolhidos (por exemplo, intervalos da grid, taxa de aprendizado, tamanho da população, número de gerações, função de fitness/perda, critérios de parada).

Realizem experimentos variando parâmetros relevantes e interpretem o comportamento observado.

Implementação e entregáveis

Linguagem: Python (todo o código deve ser escrito em Python).

Relatório (PDF): Descrição das etapas executadas, decisões de projeto, experimentos e resultados (tabelas e gráficos), análise crítica e conclusões. Incluam gráficos que mostrem o impacto de diferentes combinações de pesos.

Código-fonte: anexo ao e-mail, bem comentado e com instruções para reprodução dos experimentos.

Envio: entregar por e-mail ao professor em formato PDF, com o código anexo; identifiquem todas as submissões (relatório e anexos) com os nomes dos alunos.

Observem que a avaliação considerará tanto a qualidade da implementação quanto a justificativa das escolhas e a profundidade da análise experimental.

Prof. Luis Alvaro