Relatório detalhado

Brunella Couto, Bruno Bertolino, Guilherme de Lazari e Lucas Lima

A seguir descrevemos, passo a passo, o que foi feito no exercício presente no arquivo enviado, por que cada etapa foi escolhida, quais decisões de implementação/importantes foram tomadas, os resultados obtidos e recomendações para melhorias e reprodução. Os trechos de código e os resultados citados abaixo estão todos baseados no notebook/script que você forneceu.

1. Visão geral do problema e do anexo

O objetivo do trabalho é classificar sinais de batimento cardíaco (dataset MIT-BIH heartbeat) em 5 classes. O anexo executa três etapas principais:

1. **Análise exploratória + agrupamento não supervisionado (K-Means)** para avaliar separabilidade natural das classes.

2. **Modelagem supervisionada**: treinar uma rede neural (RNA) para classificação em 5 classes.

3. **Avaliação**: métricas por classe e matriz de confusão; discussão de desempenho e implicações clínicas.

2. Dados — carregamento e preparação

- O notebook baixa/usa o dataset **shayanfazeli/heartbeat (MIT-BIH)** do Kaggle e carrega `mitbih_train.csv` e `mitbih_test.csv` em `train_df` e `test_df`.

- Features: todas as colunas exceto a última (187 dimensões). Labels: última coluna. As shapes são impressas no notebook (treino / validação / teste).

Decisões chave:

- Conversão dos rótulos para inteiros e one-hot encoding usando `tensorflow.keras.utils.to_categorical`.

- Divisão treino/validação: `train_test_split(..., test_size=0.2, random_state=42)`.

3. Análise exploratória e K-Means (Tarefa 1)

Objetivo: ver o quanto as classes são naturalmente separáveis sem supervisão.

O notebook:

- Executa K-Means com `n_clusters=5` (igual ao nº de classes).
- Reduz dimensionalidade para 2D via PCA para visualização.
- Plota os clusters (K-Means) e, lado a lado, os rótulos reais (PCA 2D).
- Cria tabela de contingência entre clusters e rótulos reais e mostra heatmap.

Interpretação e conclusões obtidas no anexo:

- Há sobreposição entre classes no espaço de características — algumas classes (por exemplo 1 e 3) se misturam mais com outras; outras (0,2,4) são mais separáveis. Isso sugere variabilidade morfológica e que algumas classes são intrinsecamente mais difíceis.

Por que foi útil:

- K-Means não é classificador, mas fornece diagnóstico: indica se separabilidade linear/cluster-like existe; ajuda a escolher arquiteturas e o esforço de pré-processamento.

4. Modelagem supervisionada (Tarefa 2) — arquitetura e treinamento

Arquitetura usada:

- MLP (Sequential) com camadas Dense:
    - Dense(128, activation='relu', input_shape=(187,))
    - Dense(64, activation='relu')
    - Dense(32, activation='relu')

    - Dense(num_classes, activation='softmax')

- Compilação: `optimizer='adam'`, `loss='categorical_crossentropy'`, métrica `'accuracy'`.

Treinamento:

- É treinado por **10 épocas**, `batch_size=64`, com validação (20% do treino original). O histórico (`history`) é salvo.

Resultados reportados no arquivo:

- Acurácia de teste: **~97.53%**. Perda de teste em torno de 0.09. O notebook mostra também métricas por classe (precision, recall, f1) e matriz de confusão. Classes 0, 2 e 4 apresentam desempenho excelente; classes 1 e 3 apresentam recall notavelmente mais baixo (ex.: recall classe 1 ≈ 0.61; classe 3 ≈ 0.59).

5. Avaliação detalhada (Tarefa 3) — interpretação das métricas

Métricas usadas:

- Acurácia geral, precision, recall, f1-score por classe e matriz de confusão. Estas são apropriadas para problemas multiclasse; o arquivo imprime o classification_report e plota a matriz de confusão.

Observações interpretativas:

- Alta acurácia global (~97.5%) indica bom desempenho geral, porém a métrica global pode mascarar problemas em classes minoritárias ou clinicamente críticas.

- Recall baixo em classes 1 e 3 significa alta taxa de **falsos negativos** para essas classes — o modelo tende a classificá-las como outras classes. Em cenários clínicos (detecção de arritmias, por exemplo), falso negativo pode ser mais perigoso: paciente com condição não é detectado. O notebook já aborda esse ponto (resposta indicando que falso negativo é pior clinicamente).

6. Interpretação clínica e implicações de erro

- **Pior erro clinicamente: Falso Negativo** — paciente doente classificado como saudável pode não receber tratamento; risco de agravamento. O anexo destaca isso.

- **Mitigação clínica:** priorizar modelos com maior *recall* nas classes patológicas, possivelmente ajustando limiares, usando penalização de classes (class weights), ou criando um segundo estágio especializado para classes críticas.

7. Pontos fortes do trabalho entregue

- Fluxo completo: EDA, diagnóstico com K-Means, modelagem supervisionada e avaliação.

- Uso prático de PCA para visualização e tabela de contingência para quantificar correspondência cluster ↔ rótulo.

- Resultado final com acurácia muito alta (bom baseline).

8. Limitações e recomendações de melhorias (ações concretas)

Abaixo sugestões ordenadas do mais direto/impactante para o mais avançado:

1. **Usar `class_weight`** na função `model.fit` ou amostragem estratificada para combater desequilíbrio e elevar recall das classes 1 e 3.

    - Exemplo: `compute_class_weight` do sklearn e passar `class_weight` ao `fit`.

2. **Aumentar número de épocas + EarlyStopping** (monitorar `val_loss`, `patience=3–5`) para evitar underfitting/overfitting; salvar melhor modelo (`ModelCheckpoint`).

3. **Regularização**: dropout entre camadas e/ou L2 (`kernel_regularizer`) para reduzir overfitting.

4. **Normalização das features**: verificar se padronização (StandardScaler) melhora convergência; não vi no notebook — adicionar padronização pode melhorar.

5. **Arquiteturas alternativas**: testar CNNs 1D (capturam morfologia temporal do sinal), RNN/GRU/LSTM (sequência), ou híbridos; esses geralmente superam MLPs em sinais biomédicos.

6. **Augmentação de dados**: pequenas perturbações, jittering, scaling do sinal para aumentar robustez.

7. **Reamostragem / SMOTE** para classes com poucas amostras (com cuidado em sinais médicos).

8. **Calibração do modelo** (Platt scaling / isotonic) e análise de limiares por classe conforme custo clínico de FP vs FN.

9. **Interpretação do modelo**: SHAP/LIME para explicar decisões em casos críticos (importante em contexto médico).

10. **Validação cruzada estratificada** em vez de uma única divisão treino/val para estimar variância do desempenho.

11. **Ensemble**: combinar MLP com CNNs e modelos tree-based (LightGBM em features extraídas) pode melhorar robustez.

9. Como reproduzir (passo a passo curto)

1. Criar ambiente Python com dependências: `numpy pandas scikit-learn tensorflow seaborn matplotlib kagglehub` (ou adaptar para download manual do dataset).

2. Colocar `mitbih_train.csv` e `mitbih_test.csv` em `./` ou ajustar caminhos no notebook.

3. Rodar o notebook/script: ele já realiza EDA, K-Means, PCA, treino do modelo e gera relatórios/plots.

4. Para treinar com modificações (class_weight, early stopping), adaptar as chamadas `model.fit(...)` conforme recomendado acima.

5. Salvar o `history` e o `model` treinado para análise posterior.

Nota: o notebook original contém as chamadas e valores iniciais (arquitetura, epochs=10, batch_size=64, etc.), consulte o arquivo fornecido para os trechos exatos.

10. Trechos-chave e onde encontrá-los no anexo

- **Carregamento dos dados e shapes**: início do notebook (`pd.read_csv` e prints).
- **K-Means + PCA + plots**: seção “1. Análise Exploratória e Agrupamento”.
- **Split, one-hot encoding e arquitetura MLP**: seção “2. Modelagem e Classificação (Supervisionada)”.
- **Treino, avaliação e relatório**: chamadas `model.fit`, `model.evaluate`, `classification_report`, `confusion_matrix` e os plots correspondentes.

11. Conclusão

- O notebook implementa um pipeline completo para classificação de batimentos cardíacos usando MLP e fornece diagnóstico via K-Means/PCA.
- O modelo alcançou **alta acurácia (~97.5%)**, porém com recall reduzido em duas classes (1 e 3), o que exige atenção por impacto clínico (falsos negativos).
- Recomenda-se aplicar balanceamento/penalização de classes, testar arquiteturas específicas para dados temporais (CNN1D / RNN), usar early stopping e validar com cross-validation. Implementações de interpretação (SHAP) e calibração são importantes antes de qualquer uso clínico.