

Disciplina: Data Science e IA

Instituição: Cesar School

Data: 23/08/2025

Nome: Eric Cabral Neder

E-mail: ecn@cesar.school

Repositório/Entregáveis:

<https://github.com/EricCabralNeder/Tech-Lead---Dados-e-IA-2025.01/tree/main/Trabalho%20Final>

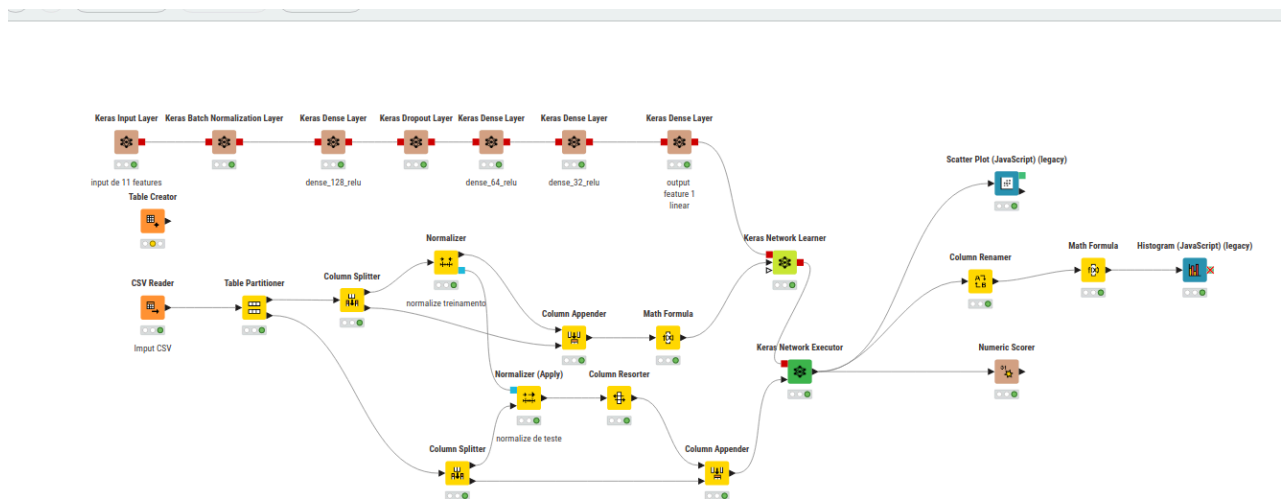
Dataset (fonte): Wine Quality – Vinho Verde (tinto) – Kaggle

<https://www.kaggle.com/datasets/joebeachcapital/wine-quality>

Licença: CC BY 4.0

Objetivo: Construir, no KNIME, um modelo de **regressão com redes neurais** (Keras/TensorFlow) para prever a variável **quality** (0–10) a partir de **11 atributos** físico-químicos de vinhos tintos.

Workflow no KNIME (visão geral)



Passo a passo dos nós (workflow)

1. **CSV Reader** — Leitura do `winequality-red.csv` (1599×12), delimitador `;`, *header* ligado, tipos numéricos.
2. **Table Partitioner** — Divisão treino (70%) e teste (30%), **seed=42** (reprodutibilidade).
3. **Column Splitter (treino)** — Separa **11 features** (inputs físico-químicos) e **alvo quality**.
4. **Math Formula** — Garante **quality** como **Double** (`$quality$ * 1.0`) para a tarefa de regressão no Keras.
5. **Normalizer (Train)** — **Z-Score** nas 11 features **apenas no treino**; gera **PMML**.
6. **Keras Input Layer** — Entrada com **shape=[11]**.
7. **Keras Batch Normalization** — BN após a entrada (estabiliza o treino).
8. **Keras Dense (128, ReLU)** — Primeira camada densa.
9. **Keras Dropout (~10%)** — Regularização entre 128→64 (*keep prob* ≈ 0.9).
10. **Keras Dense (64, ReLU)** — Segunda camada densa.
11. **Keras Dense (32, ReLU)** — Terceira camada densa.
12. **Keras Dense (1, Linear)** — Saída escalar (regressão).
13. **Keras Network Learner** — **Loss: MAE**; **Adam lr=0.0005**; **batch=32**; **épocas=800**; *shuffle ON*; **seed=42**.
14. **Column Splitter (teste)** — Separa features e **quality** no conjunto de teste.
15. **Normalizer (Apply)** — Aplica o **PMML do treino** no teste (evita vazamento).
16. **Column Resorter** — Garante **ordem idêntica** das 11 features entre treino e teste.
17. **Column Appender (teste)** — Reanexa **quality** real ao conjunto preparado.
18. **Keras Network Executor** — Gera **prediction(quality)** mantendo as colunas de entrada.
19. **Column Renamer / Math Formula (opcional)** — Renomeia predição para **y_pred**; cria **residual = quality - y_pred**.

20. **Numeric Scorer** — Métricas de regressão (R^2 , RMSE, MAE, MSE, MAPE, MSD).

21. **Scatter Plot (JavaScript)** — Gráfico de dispersão `quality` vs `prediction(quality)`.

22. **Histogram (JavaScript)** — Histograma do *target* (ou de resíduos, se preferir).

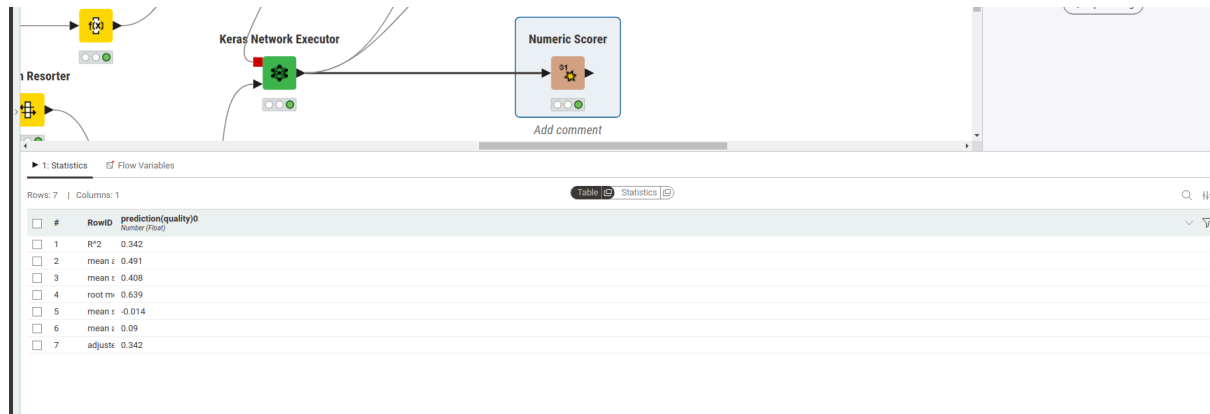
Métricas de performance (conjunto de teste)

- $R^2 = 0,342$
- $RMSE = 0,639$
- $MAE \approx 0,491$
- $MSE \approx 0,408$
- $MAPE \approx 0,09$
- **Mean signed difference $\approx -0,014$**
- **Baseline (prever média): $RMSE \approx 0,807$ (*pior que o modelo*)**

Interpretação breve: O modelo explica ~34% da variância de *quality* e reduz o erro médio (RMSE/MAE) em relação ao baseline de prever a média. O viés médio é próximo de zero.

Análise de performance (conjunto de teste)

Métricas no conjunto de teste (Numeric Scorer)

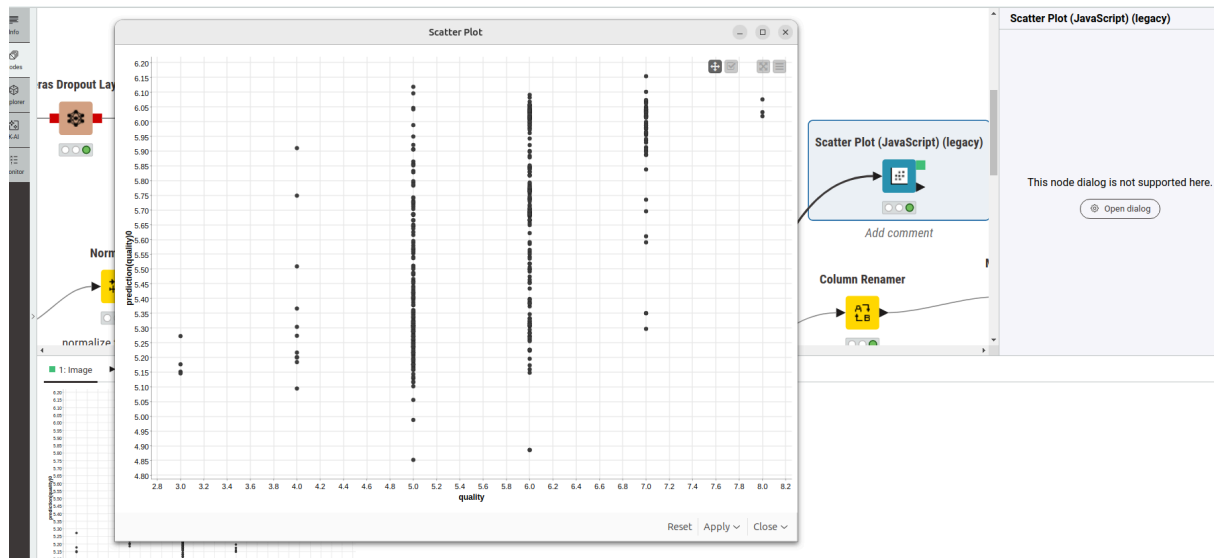


O que mostra: Tabela com as métricas de regressão calculadas no KNIME.

Leitura/explicação:

- **$R^2 = 0,342$ — o modelo explica cerca de 34% da variância de *quality*, um ganho substancial sobre o baseline.**
- **$RMSE = 0,639$ e $MAE \approx 0,491$ — erros médios na mesma escala de *quality* (0–10); quanto menores, melhor.**
- **Baseline (prever média) teria $RMSE \approx 0,807$ (pior).**
- **Mean signed difference $\approx -0,014$ — viés médio muito próximo de zero (tendência não enviesada).**

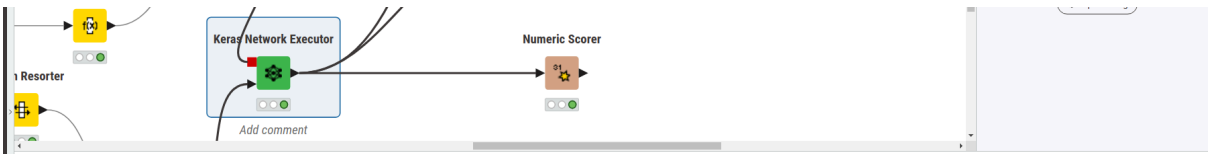
Dispersão: quality (x) vs. prediction(quality) (y)



O que mostra: cada ponto é uma amostra do teste, com o valor real de *quality* no eixo X e a predição no eixo Y.

Leitura/explicação: quanto mais próximos da diagonal ($y=x$), melhores as predições. A nuvem concentrada em torno da diagonal indica ajuste razoável; observa-se leve “regressão à média” (extremos tendem a ser sub/superestimados), coerente com o valor de R^2 .

Amostras do conjunto de teste (predições do Keras)



1: Data Table

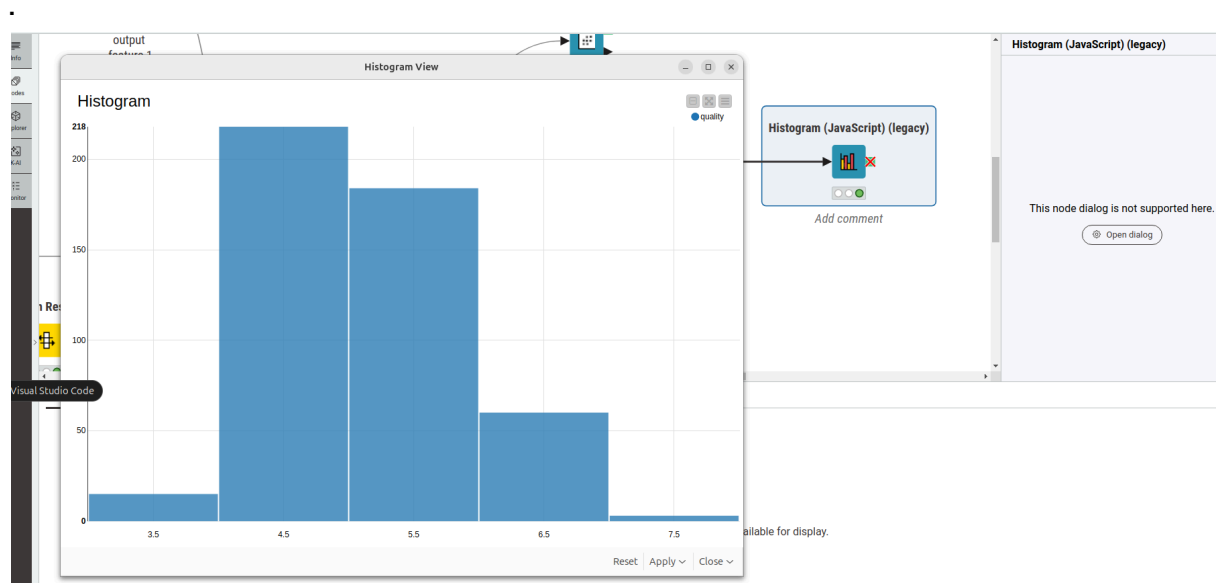
Rows: 480 | Columns: 13

| # | RowID | fixed acid... Number (float) | volatile acidity Number (float) | citric acid Number (float) | residual sugar Number (float) | chlorides Number (float) | free sulfur di... Number (float) | total sulfur di... Number (float) | density Number (float) | pH Number (float) | sulphates Number (float) | alcohol Number (float) | quality Number (integer) | prediction(qu... Number (float) |
|----|-------|---------------------------------|------------------------------------|-------------------------------|----------------------------------|-----------------------------|-------------------------------------|--------------------------------------|---------------------------|----------------------|-----------------------------|---------------------------|-----------------------------|------------------------------------|
| 1 | Row4 | -0.545 | 0.979 | -1.398 | -0.47 | -0.247 | -0.463 | -0.363 | 0.567 | 1.299 | -0.58 | -0.976 | 5 | 5.26 |
| 2 | Row7 | -0.504 | 0.698 | -1.398 | -0.991 | -0.459 | -0.081 | -0.752 | -1.179 | 0.517 | -1.136 | -0.395 | 7 | 5.297 |
| 3 | Row9 | -0.486 | -0.144 | 0.445 | 2.653 | -0.343 | 0.11 | 1.672 | 0.567 | 0.257 | 0.9 | 0.089 | 5 | 5.4 |
| 4 | Row16 | 0.105 | -1.38 | 1.469 | -0.545 | 0.061 | 1.829 | 1.702 | 0.076 | -0.069 | 0.592 | 0.089 | 7 | 5.735 |
| 5 | Row22 | -0.249 | -0.537 | -0.323 | -0.693 | 0.33 | -0.559 | -0.273 | -0.087 | -0.916 | 1.579 | -0.879 | 5 | 5.724 |
| 6 | Row23 | 0.105 | -0.2 | -0.835 | -0.173 | -0.093 | -0.654 | 0.625 | 0.022 | -0.916 | -0.766 | -0.976 | 5 | 5.405 |
| 7 | Row25 | -1.194 | -0.762 | -0.579 | -0.842 | -0.17 | -0.463 | -0.692 | -0.688 | 0.191 | -0.58 | -1.073 | 5 | 5.65 |
| 8 | Row27 | -0.249 | -0.537 | -0.323 | -0.693 | 0.33 | -0.559 | -0.273 | -0.087 | -0.916 | 1.579 | -0.879 | 5 | 5.724 |
| 9 | Row30 | -0.958 | 0.838 | -1.04 | -0.098 | 0.003 | 0.11 | 1.073 | -0.524 | 0.257 | -0.704 | -0.298 | 5 | 5.303 |
| 10 | Row34 | -1.844 | -1.155 | -0.118 | -0.545 | 0.273 | -0.272 | 0.116 | -0.579 | 0.452 | -0.642 | -1.17 | 5 | 5.466 |
| 11 | Row45 | -2.198 | -0.032 | -0.63 | -0.322 | -0.67 | -0.75 | 0.565 | -1.834 | 3.84 | -0.58 | 2.606 | 4 | 5.749 |

O que mostra: um recorte tabular com features normalizadas, a quality real e a prediction(quality) gerada pela rede.

Leitura/explicação: serve para inspecionar exemplos individuais, checar que a coluna de predição foi anexada corretamente (Executor com *Keep input columns*) e comparar, linha a linha, a proximidade entre real e previsto

Histograma de *quality* (distribuição do alvo)



O que mostra: distribuição de frequências da variável *quality* no dataset.

Leitura/explicação: a maior concentração em notas intermediárias (p. ex., 5–6) explica a dificuldade de capturar extremos e por que o modelo tende a “puxar” previsões para a média; essa distribuição também contextualiza o valor de R^2 obtido.

Conclusão (processo, dificuldades e ajustes)

Neste projeto, construí um pipeline completo no KNIME para prever a variável *quality* do **Wine Quality (Vinho Verde tinto)** a partir de 11 medidas físico-químicas. O fluxo inclui leitura do CSV, particionamento 70/30 (seed fixa), normalização **Z-Score** calculada **apenas no treino** (PMML) e aplicada no teste (evitando *data leakage*), organização da ordem das colunas (Column Resorter) e modelagem com **Keras** (BN → Dense(128) → Dropout(~10%) → Dense(64) → Dense(32) → Dense(1), loss=MAE, Adam lr≈0,0005, batch=32, ~800 épocas). As métricas no conjunto de teste foram **$R^2 = 0,342$** , **RMSE = 0,639** e **MAE ≈ 0,491**, superando o **baseline** de prever a média (**RMSE ≈ 0,807**). Os gráficos de dispersão e histograma indicam nuvem próxima à diagonal e alvo concentrado em notas medianas (desafio para capturar extremos). **Objetivo cumprido**: o workflow é reproduzível, modular e melhora claramente o baseline.

Dificuldades encontradas e como resolvi

- **Diferenças de UI no KNIME 5.x**: alguns nós/referências presentes em materiais antigos (ex.: *Reference Column Filter* ou “Switch to Classic UI”) não apareciam na versão moderna.
Solução: adaptei o pipeline usando apenas nós disponíveis na UI atual (ex.: marcar **Keep input columns** no *Keras Network Executor* e garantir a saída do último *layer*), além de checar conexões (triângulo = PMML/modelo, quadrado = tabela) para não misturar portas.
- **Erro ao criar workflow (“Problem fetching space items”) e exportação (“Selection contains no elements”)**: problemas relacionados ao *Explorer* e ao ponto de montagem do workspace.
Solução: montar/selecionar o **LOCAL (workspace)** nas preferências do KNIME Explorer; ao exportar, **selecionar o workflow raiz** no diálogo (*Select...*) e salvar como **.knwf**. Como *fallback*, compactar a pasta do workflow e renomear para **.knwf**.
- **Tipo do alvo e colunas de entrada**: o **quality** estava numérico, mas o Keras exigiu consistência de tipo/ordem.
Solução: **Math Formula** para garantir **quality** como **Double** quando necessário e **Column Resorter** para padronizar a ordem exata das 11 *features* entre treino e teste.

- **Vazamento de dados (normalização):** risco de calcular estatísticas nos dados de teste.
Solução: **Normalizer (Train)** só no treino (gerando PMML) + **Normalizer (Apply)** no teste.
- **Modelo inicialmente “colando” na média ($R^2 \leq 0$):** a rede não aprendia variação suficiente.
Solução: incluir **Batch Normalization** na entrada, **Dropout (~10%)** e *tuning* de **learning rate** ($\approx 0,0005$), *batch size* (32) e **épocas** (~800). Regularização L2 não estava disponível diretamente na UI, então priorizei BN + Dropout.
- **Distribuição do alvo desbalanceada** (muitas notas 5–6): dificulta previsões em extremos e reduz teto de R^2 .
Solução: aceitar a limitação intrínseca do dataset e avaliar o modelo contra um baseline forte (prever a média), demonstrando ganho real (redução de RMSE/MAE).