**Comparação de Modelos Clássicos de Aprendizado Supervisionado na Classificação de Desempenho Escolar**

**1 - Objetivo**

Este projeto tem como principal objetivo analisar, comparar e avaliar o desempenho de diferentes algoritmos de aprendizado supervisionado na tarefa de classificação binária de desempenho escolar, utilizando um conjunto de dados real, extraído do UCI Machine Learning Repository.

A avaliação é realizada tanto do ponto de vista preditivo quanto da interoperabilidade, considerando diferentes métricas, tratamento de desbalanceamento, estratégias de pré-processamento de variáveis categóricas, normalização, assim como o treinamento de modelos de complexidade variada.

Este pipeline proporciona uma visão comparativa e fundamentada para apoiar a seleção do modelo na resolução de um problema prático de análise da eficácia do ensino.

**2 - Conjunto de Dados (Dataset)**

* Fonte: [*UCI Machine Learning Repository — Student Performance*](https://archive.ics.uci.edu/ml/datasets/Student+Performance)
* Arquivo: student-mat.csv
* Quantidade de instâncias: 395
* Quantidade de atributos: 33
* Natureza dos dados: Demográficos, socioeconômicos, pessoais e relacionados ao desempenho.
* Variável-Alvo (Target): pass (criada pelo pipeline) — sendo:
  + 1 para aprovado (nota final G3 maior ou igual a 10).
  + 0 para reprovado (nota G3 menor que 10).

**3 - Pipeline do Projeto**

Este pipeline cobre o ciclo fim a fim de um modelo de aprendizado de máquina, sendo ele:

* Coleta e tratamento de dados
* Análise exploratória
* Engenharia de atributos
* Treinamento de diferentes modelos
* Avaliação comparativa
* Análise de resultados e definição de caminhos futuros

**3.1 Importação de Bibliotecas**

Foram utilizadas as seguintes bibliotecas:

* Manipulação de Dados: pandas e numpy.
* Visualização: matplotlib e seaborn.
* Modelagem: scikit-learn (árvore de decisão, floresta aleatória, métricas) e tensorflow.keras.
* Pré-Processamento: imbalanced-learn (para SMOTE) e scikit-learn (OneHotEncoder, StandardScaler).

**3.2 Leitura e Pré-Processamento dos Dados**

Primeiro é realizado o load do conjunto de dados a partir do CSV:

*file\_path = './dataset/student-mat.csv'*

*df = pd.read\_csv(file\_path, sep=';')*

**3.2.1 Criação da variável-Alvo (Target)**

*df['pass'] = (df['G3'] >= 10).astype(int)*

* 1 = Aprovado
* 0 = Reprovado

**3.3 Análise Exploratória de Dados (EDA)**

* Distribuição da variável-Alvo:

*sns.countplot(data=df, x='pass')*

*plt.title("Distribuição da Aprovação")*

*plt.xlabel("Pass")*

*plt.ylabel("Quantidade")*

*plt.show()*

* Identificação de Atributos Categóricos:

*cat\_cols = df.select\_dtypes(include='object').columns*

*print("Colunas categóricas :", cat\_cols)*

* Uma análise univariada e multivariada é importante para:
  + Detecção de outliers
  + Correlações entre variáveis
  + Dependências que serão levadas em conta na hora de criar o modelo

**3.4 Pré-Processamento**

* Codificação de Categóricos (One-Hot-Encoding):

*df = pd.get\_dummies(df, columns=cat\_cols, drop\_first=True)*

* Separação das Features (X) e do Alvo (Y):

*X = df.drop(['G1', 'G2', 'G3', 'pass'], axis=1)*

*y = df['pass']*

* Normalização (Padronização) das Variáveis:

*scaler = StandardScaler()*

*X\_scaled = scaler.fit\_transform(X)*

* Balanceamento das Classes (SMOTE) para tratamento do desbalanceamento:

*smote = SMOTE(random\_state=42)*

*X\_res, y\_res = smote.fit\_resample(X\_scaled, y)*

**3.5 Divisão dos Dados**

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(*

*X\_res, y\_res, test\_size=0.3, random\_state=42*

*)*

* Dados de treinamento: 70%
* Dados de teste: 30%
* Random state: 42 (garante a reprodutibilidade)

**3.6 Modelagem**

**3.6.1 Árvore de Decisão (Decision Tree)**

*dt = DecisionTreeClassifier(random\_state=42)*

*dt.fit(X\_train, y\_train)*

*y\_pred\_dt = dt.predict(X\_test)*

*print("Relatório - Árvore de Decisão:")*

*print(classification\_report(y\_test, y\_pred\_dt))*

**3.6.2 Random Forest**

*rf = RandomForestClassifier(n\_estimators=100, random\_state=42)*

*rf.fit(X\_train, y\_train)*

*y\_pred\_rf = rf.predict(X\_test)*

*print("Relatório - Random Forest:")*

*print(classification\_report(y\_test, y\_pred\_rf))*

**3.6.3 Rede Neural Artificial (Multi-layer Perceptron)**

* Arquitetura da rede:
  + Uma camada densa com 32 neurônios (ativação ReLU)
  + Uma camada de dropout (Dropout = 0.3) para prevenir overfitting
  + Uma nova camada densa com 16 neurônios (ativação ReLU)
  + Uma saída densa com 1 neurônio (ativação Sigmóide)
* Compilação:
  + Loss: binary\_crossentropy
  + Otimizador: adam
  + Métrica: accuracy

*model = Sequential([*

*Dense(32, input\_dim=X\_train.shape[1], activation='relu'),*

*Dropout(0.3),*

*Dense(16, activation='relu'),*

*Dense(1, activation='sigmoid')*

*])*

*model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])*

*model.fit(X\_train, y\_train, epochs=20, batch\_size=32, verbose=1)*

*y\_pred\_nn\_prob = model.predict(X\_test)*

*y\_pred\_nn = (y\_pred\_nn\_prob > 0.5).astype(int)*

*print("Relatório - Rede Neural Artificial:")*

*print(classification\_report(y\_test, y\_pred\_nn))*

**3.7 Avaliação e Análise Comparativa**

* Relatório de Classificação
* Matriz de Confusão
* Curvas ROC e AUC

*models = ['Decision Tree', 'Random Forest', 'Neural Network']*

*preds = [y\_pred\_dt, y\_pred\_rf, y\_pred\_nn]*

*probs = [dt.predict\_proba(X\_test)[:, 1],*

*rf.predict\_proba(X\_test)[:, 1],*

*y\_pred\_nn\_prob.flatten()]*

*for model\_name, pred in zip(models, preds):*

*print(f"{model\_name} AUC: {roc\_auc\_score(y\_test, pred):.4f}")*

*cm = confusion\_matrix(y\_test, pred)*

*sns.heatmap(cm, annot=True, fmt='d')*

*plt.title(f'Matriz de Confusão - {model\_name}')*

*plt.xlabel('Valor Previsto')*

*plt.ylabel('Valor Real')*

*plt.show()*

*plt.figure(figsize=(10, 6))*

*for model\_name, probas in zip(models, probs):*

*fpr, tpr, \_ = roc\_curve(y\_test, probas)*

*auc = roc\_auc\_score(y\_test, probas)*

*plt.plot(fpr, tpr, label=f'{model\_name} (AUC = {auc:.2f})')*

*plt.plot([0, 1], [0, 1], 'k--')*

*plt.xlabel('False Positive Rate')*

*plt.ylabel('True Positive Rate')*

*plt.title('Curvas ROC dos Modelos')*

*plt.legend()*

*plt.grid()*

*plt.show()*

**4 - Boas Práticas e Pontos Fortes**

* Pipeline claro, coeso e facilmente reproduzível.
* Sementes aleatórias fixadas para proporcionar reprodutibilidade.
* Pré-processamento realizado de forma consistente para todos os modelos, aumentando a comparabilidade.
* Visualizações que permitem tanto uma avaliação numérica quanto gráfica.
* Comparação junto às métricas de desempenho, aumentando a eficácia na seleção do modelo.

**5 - Conclusão**

Este pipeline proporciona uma visão completa, profunda e profissional do uso de aprendizado de máquina na análise de desempenho escolar.  
Com ele, é possível:

* Realizar um tratamento de dados robusto.
* Aplicar diferentes algoritmos de aprendizado supervisionado.
* Comparar suas performances de forma justa e fundamentada.
* Implementar um modelo que ajude na compreensão dos mecanismos que influenciam o desempenho dos estudantes.

**6 - Referências**

* [*UCI Machine Learning Repository — Student Performance*](https://archive.ics.uci.edu/ml/datasets/Student+Performance)
* [*Scikit-learn — Documentação*](https://scikit-learn.org/)
* [*TensorFlow/Keras — Documentação*](https://www.tensorflow.org/)
* [*Imbalanced-learn — SMOTE*](https://imbalanced-learn.org/)