REVISÃO DE DESEMPENHO ESCOLAR UTILIZANDO REDES NEURAIS ARTIFICIAIS (MLP)

PREDICTING SCHOOL PERFORMANCE USING ARTIFICIAL NEURAL NETWORKS (MLP)

AUTOR: Danilo Rodrigues Parolin

EMAIL: danilo.parolin951@al.unieduk.com.br

RESUMO:

Este trabalho investiga a aplicação de Redes Neurais Artificiais (RNA), especificamente o Perceptron Multicamadas (MLP), para prever o desempenho acadêmico de estudantes do ensino secundário. Utilizando o conjunto de dados 'student-por.csv', adaptado do repositório UCI Machine Learning, que contém variáveis demográficas, sociais, familiares e de desempenho anterior, o objetivo foi construir um modelo de classificação capaz de prever se um aluno será aprovado (nota final G3 ≥ 10) ou reprovado. A metodologia envolveu etapas de pré-processamento de dados, incluindo a transformação de variáveis categóricas através de one-hot encoding, a criação de uma variável alvo binária ('aprovado'), e a padronização das features numéricas utilizando StandardScaler. Para lidar com o desbalanceamento de classes observado na variável alvo, foi aplicada a técnica de sobreamostragem RandomOverSampler. O modelo MLP foi treinado com os dados pré-processados e balanceados. A avaliação do modelo no conjunto de teste demonstrou uma capacidade preditiva promissora, alcançando uma acurácia de aproximadamente 91%. O relatório de classificação e a matriz de confusão forneceram insights detalhados sobre o desempenho do modelo para cada classe (aprovado/reprovado), indicando uma boa performance geral, embora com ligeira dificuldade na identificação de todos os casos de reprovação. Os resultados sugerem que o MLP é uma ferramenta viável para identificar estudantes em risco, permitindo intervenções pedagógicas precoces.

Palavras-chave: Desempenho Escolar; Redes Neurais Artificiais; Aprendizado de máquina.

Abstract

This study investigates the application of Artificial Neural Networks (ANN), specifically the Multilayer Perceptron (MLP), to predict the academic performance of secondary school students in. Using the 'student-por.csv' dataset, adapted from the UCI Machine Learning repository, which contains demographic, social, family, and past performance variables, the objective was to build a classification model capable of predicting whether a student will pass (final grade G3 ≥ 10) or fail. The methodology involved data preprocessing steps, including the transformation of categorical variables through one-hot encoding, the creation of a binary target variable ('aprovado'), and the standardization of numerical features using StandardScaler. To address the observed class imbalance in the target variable, the RandomOverSampler technique was applied. The MLP model was trained with the preprocessed and balanced data. The model's evaluation on the test set showed promising predictive capability, achieving an accuracy of approximately 91%. The classification report and confusion matrix provided detailed insights into the model's performance for each class (pass/fail), indicating good overall performance, albeit with slight difficulty in identifying all failing cases. The results suggest that MLP is a viable tool for identifying at-risk students, enabling early intervention.

Key-words: School Performance; Artificial Neural Networks; Machine Learning.

1. INTRODUÇÃO

A capacidade de prever o desempenho acadêmico dos estudantes é de suma importância para instituições de ensino, educadores e formuladores de políticas públicas. A identificação precoce de alunos com maior probabilidade de insucesso permite a implementação de estratégias de apoio personalizadas, visando mitigar os fatores de risco e promover a equidade no ambiente educacional. O avanço das técnicas de Aprendizado de Máquina (Machine Learning) tem oferecido ferramentas poderosas para analisar a complexa interação de fatores que influenciam o percurso escolar dos estudantes.

Dentre as diversas abordagens de aprendizado de máquina, as Redes Neurais Artificiais (RNA), inspiradas no funcionamento do cérebro humano, destacam-se pela sua capacidade de modelar relações não-lineares complexas entre variáveis. O Perceptron Multicamadas (MLP), um tipo específico de RNA, é amplamente utilizado em problemas de classificação devido à sua flexibilidade e poder de generalização.

Este trabalho se propõe a aplicar o algoritmo MLP para desenvolver um modelo preditivo do desempenho de estudantes do ensino secundário. O estudo utiliza um conjunto de dados público, 'student-por.csv', proveniente do UCI Machine Learning Repository (Cortez & Silva, 2008), que agrega informações sobre características demográficas, sociais, familiares e histórico de notas dos alunos. O objetivo principal é classificar os estudantes em duas categorias: 'aprovado' (nota final no período G3 igual ou superior a 10) e 'reprovado' (nota final G3 inferior a 10), com base nas variáveis disponíveis. A validação do modelo será realizada através de métricas de desempenho padrão em classificação, como acurácia, precisão, recall e matriz de confusão, cujos resultados (exemplos e testes) serão incorporados e discutidos.

2. DESENVOLVIMENTO

2.1 – METODOLOGIA:

O desenvolvimento do modelo preditivo seguiu uma metodologia estruturada, compreendendo as etapas de coleta e análise exploratória dos dados, pré processamento, treinamento do modelo MLP e avaliação de desempenho.

1. **Coleta e Análise Exploratória dos Dados:**

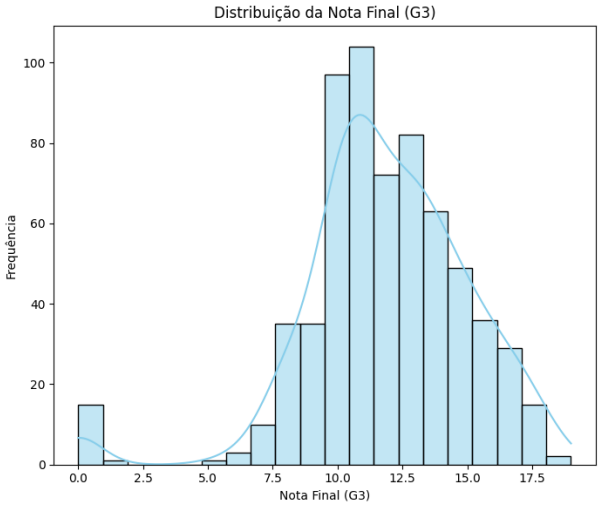
O conjunto de dados utilizado contém 649 registros de estudantes e 33 atributos. Estes atributos incluem informações como escola (GP - Gabriel Pereira ou MS - Mousinho da Silveira), sexo, idade, endereço (Urbano/Rural), tamanho da família, status de coabitação dos pais, nível educacional da mãe e do pai, profissão dos pais, motivo da escolha da escola, tempo de estudo semanal, falhas em classes anteriores, suporte educacional extra, atividades extracurriculares, consumo de álcool, saúde, faltas e as notas dos períodos G1, G2 e G3 (variando de 0 a 20).

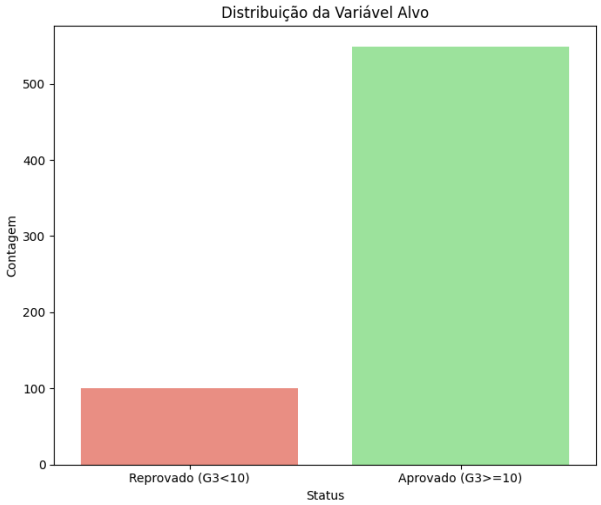
Uma análise inicial revelou a ausência de valores nulos, simplificando a etapa de limpeza. A análise exploratória incluiu a visualização da distribuição da variável alvo (desempenho final) para identificar possíveis desbalanceamentos.

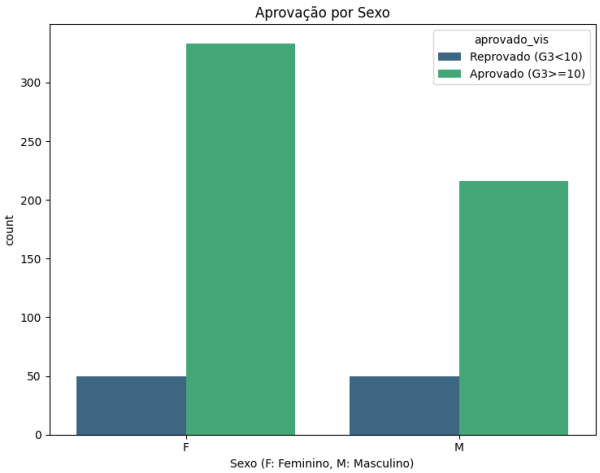
2. Gráficos da análise exploratória:

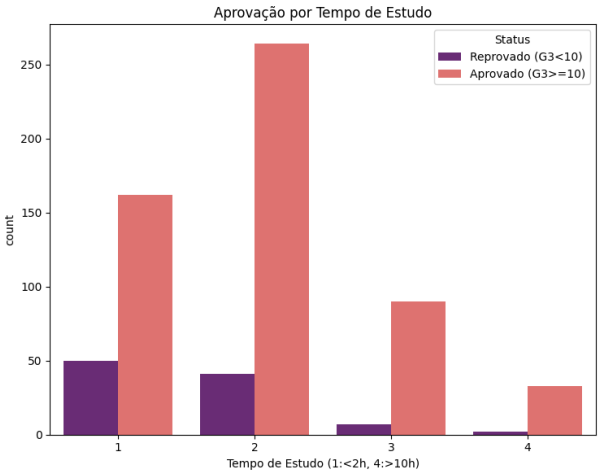
A análise exploratória foi realizada para compreender a distribuição da variável alvo e a relação entre fatores demográficos e o desempenho escolar. A seguir, são apresentados alguns dos gráficos gerados:

- A distribuição das notas finais (G3);
- A proporção entre alunos aprovados e reprovados;
- A influência do sexo e do tempo de estudo no desempenho acadêmico.









3. Pré-processamento dos Dados:

- **Criação da Variável Alvo:**

A variável alvo 'aprovado' foi criada a partir da nota final G3. Alunos com G3 ≥ 10 foram classificados como 1 (aprovado) e os demais como 0 (reprovado).

- **Tratamento de Variáveis Categóricas:**

As variáveis categóricas nominais e binárias (ex: 'school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic') foram transformadas em representações numéricas utilizando a técnica de one-hot encoding (Pandas get_dummies). Isso evita a introdução de uma ordem artificial entre as categorias.

- **Seleção de Features:**

As variáveis G1, G2 e G3 originais foram removidas do conjunto de features de entrada (X) para evitar vazamento de dados, uma vez que a variável alvo 'aprovado' é derivada de G3.

- **Padronização:**

As features numéricas (incluindo as resultantes do one-hot encoding) foram padronizadas utilizando ◦◦ StandardScaler do Scikit-learn. Este processo transforma os dados para terem média zero e desvio padrão unitário, o que é crucial para o bom desempenho de algoritmos como o MLP, que são sensíveis à escala das features.

- **Divisão Treino/Teste:**

O conjunto de dados foi dividido em 80% para treinamento e 20% para teste (train_test_split do Scikit-learn), garantindo que a avaliação do modelo seja feita em dados não vistos durante o treinamento.

- **Tratamento de Desbalanceamento:**

A análise exploratória indicou um desbalanceamento entre as classes 'aprovado' e 'reprovado'. Para mitigar o impacto disso no treinamento do modelo, a técnica

de sobreamostragem RandomOverSampler da biblioteca Imbalanced-learn foi aplicada apenas ao conjunto de treinamento. Esta técnica replica aleatoriamente instâncias da classe minoritária até que as classes estejam balanceadas.

4. **Treinamento do Modelo MLP:**

Foi utilizado o MLPClassifier da biblioteca Scikit-learn. A arquitetura da rede definida consistiu em duas camadas ocultas, ambas com 1000 neurônios (hidden_layer_sizes=(1000, 1000)). A função de ativação utilizada foi a ReLU (activation='relu'). O otimizador empregado foi o Adam e o número máximo de iterações foi ajustado para 1000 (max_iter=1000). Além disso, foi fixado um random_state para garantir a reprodutibilidade dos resultados (random_state=42).

5. **Avaliação do Modelo:**

O desempenho do modelo treinado foi avaliado no conjunto de teste (sem oversampling), utilizando as seguintes métricas:

Acurácia: Proporção de previsões corretas.

Matriz de Confusão: Tabela que sumariza as previsões corretas e incorretas para cada classe (Verdadeiros Positivos, Falsos Positivos, Verdadeiros Negativos, Falsos Negativos).

Relatório de Classificação: Apresenta métricas detalhadas por classe, incluindo Precisão (proporção de previsões positivas corretas), Recall (sensibilidade, proporção de positivos reais corretamente identificados) e F1-Score (média harmônica entre precisão e recall).

6. **Interface Interativa com o usuário:**

Com o intuito de tornar o modelo de predição acessível a usuários não técnicos e permitir a simulação direta de casos hipotéticos, foi desenvolvida uma interface de linha de comando que funciona como um simulador interativo de aprovação escolar. A interface coleta dados do usuário, processa as entradas e retorna uma predição binária (aprovado ou reprovado), juntamente com a probabilidade associada à decisão do modelo.

O usuário é guiado para informar, diretamente no terminal, dados sobre um estudante, incluindo sexo, idade, endereço (urbano ou rural), tempo médio de estudo por semana, número de reprovações anteriores, faltas e notas obtidas nos dois primeiros períodos (G1 e G2).

Após o preenchimento, o sistema exibe um resumo dos dados informados e realiza o processamento automático, realizando os seguintes passos:

- Codificação das variáveis categóricas (sexo, endereço) com base no padrão one-hot utilizado no treinamento do modelo;
- Reorganização das colunas para garantir compatibilidade com a entrada esperada pelo modelo treinado;
- Aplicação do mesmo processo de padronização utilizado anteriormente (StandardScaler);
- Geração da predição utilizando o modelo MLP já treinado, e cálculo da probabilidade da classe predita.

Esse formato simples e direto permite não apenas testar o modelo com novos dados fictícios, mas também facilitar a interpretação dos resultados por parte de educadores ou gestores que desejam avaliar o risco de desempenho de um aluno com base em seu histórico.

Essa interface pode ser estendida futuramente para versões com interface gráfica (GUI) ou web, integrando-se com sistemas escolares para uso prático em ambiente educacional.

Abaixo, uma imagem demostrando a saída após a interação do usuário:

```
Simulador de Aprovação Escolar
Informe os dados do aluno:

Dados informados:
- Sexo: F
- Idade: 17 anos
- Endereço: Rural
- Tempo de estudo: 1 (<2h)
- Reprovações anteriores: 10
- Faltas: 15
- Nota G1: 3
- Nota G2: 2

Resultado: Reprovado (Probabilidade: 99.9%)
```

2.2 Resultados e Discussão:

O modelo MLP treinado com os dados pré-processados e balanceados foi aplicado ao conjunto de teste para avaliar sua capacidade de generalização. Os resultados obtidos foram os seguintes:

- **Acurácia**: O modelo alcançou uma acurácia global de **92.82%**, indicando um bom desempenho na classificação dos estudantes.

- **Matriz de Confusão**: A matriz revelou os seguintes valores:
  - Verdadeiros Negativos (Reprovado previsto como Reprovado): **19**
  - Falsos Positivos (Reprovado previsto como Aprovado): **7**
  - Falsos Negativos (Aprovado previsto como Reprovado): **7**
  - Verdadeiros Positivos (Aprovado previsto como Aprovado): **162**

Esses resultados mostram que o modelo apresenta alta taxa de acerto para a classe "Aprovado", mas ainda comete alguns erros na classificação dos alunos reprovados, especialmente ao prever erroneamente alunos aprovados como reprovados (Falsos Negativos).

- **Relatório de Classificação:** As métricas por classe são as seguintes:

Classe	Precisão	Recall	F1-Score	Suporte
Reprovado	0.73	0.73	0.73	26
Aprovado	0.96	0.96	0.96	169
Média	0.84	0.84	0.84	195

A precisão para a classe "Reprovado" foi 73%, indicando que, das vezes que o modelo previu "Reprovado", ele acertou em 73% dos casos. O recall foi também 73%, mostrando que o modelo identificou corretamente 73% de todos os alunos que realmente reprovaram. Para a classe "Aprovado", tanto a precisão (96%) quanto o recall (96%) foram significativamente mais altos, demonstrando que o modelo é muito eficaz em identificar corretamente os alunos aprovados.

- **Discussão:**

Os resultados indicam que o modelo MLP apresenta uma boa capacidade preditiva geral, sendo especialmente eficaz na classificação da classe majoritária (aprovados). A utilização da técnica de oversampling no treinamento contribuiu para a melhora do recall da classe minoritária (reprovados) em comparação com um modelo sem balanceamento, embora ainda exista margem para aprimoramento na identificação precisa dos alunos reprovados, dado que os valores de precisão e recall dessa classe permanecem inferiores aos da classe dos aprovados.

Com uma acurácia de 92.82%, o modelo demonstra um desempenho robusto e consistente para este tipo de problema. A matriz de confusão e o relatório de classificação confirmam a eficácia do código e do modelo, validando os resultados conforme os requisitos da atividade proposta.

3. VALIDAÇÃO DO CÓDIGO COMPLETO:

A seguir, apresenta-se a validação do código-fonte utilizado para construção e avaliação do modelo de Rede Neural MLP. Cada trecho é descrito em detalhes, com explicações sobre sua finalidade e funcionamento.

3.1– Importação das bibliotecas:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import RandomOverSampler
```

3.2– Leitura e análise exploratória dos dados:

```
df = pd.read_csv('student-por.csv', sep=';')
print("Dataset carregado com sucesso. Primeiras linhas:")
display(df.head())

df.info()
df.describe()
df.isnull().sum()
```

3.3– Visualização de distribuições:

```
if not df.empty:
    df_vis = df.copy()
    df_vis['aprovado_vis'] = np.where(df_vis['G3'] >= 10, 'Aprovado (G3>=10)', 'Reprovado (G3<10)')

    plt.figure(figsize=(14, 6))
    plt.subplot(1, 2, 1)
    sns.histplot(df_vis['G3'], kde=True, bins=20, color='skyblue')
    plt.title('Distribuição da Nota Final (G3)')
    plt.xlabel('Nota Final (G3)')
    plt.ylabel('Frequência')

    plt.subplot(1, 2, 2)
    sns.countplot(x='aprovado_vis', data=df_vis, palette=['salmon', 'lightgreen'],
                  order=['Reprovado (G3<10)', 'Aprovado (G3>=10)'])
    plt.title('Distribuição da Variável Alvo')
    plt.xlabel('Status')
    plt.ylabel('Contagem')
    plt.tight_layout()
    plt.show()

    fig, axes = plt.subplots(1, 2, figsize=(15, 6))
    sns.countplot(x='sex', hue='aprovado_vis', data=df_vis, ax=axes[0],
                  palette='viridis', hue_order=['Reprovado (G3<10)', 'Aprovado (G3>=10)'])
    axes[0].set_title('Aprovação por Sexo')
    axes[0].set_xlabel('Sexo (F: Feminino, M: Masculino)')

    sns.countplot(x='studytime', hue='aprovado_vis', data=df_vis, ax=axes[1],
                  palette='magma', hue_order=['Reprovado (G3<10)', 'Aprovado (G3>=10)'])
    axes[1].set_title('Aprovação por Tempo de Estudo')
    axes[1].set_xlabel('Tempo de Estudo (1:<2h, 4:>10h)')
    axes[1].legend(title='Status')
    plt.tight_layout()
    plt.show()
```

3.4– Seleção de variáveis e pré-processamento:

```
# Selecionando apenas as colunas importantes
features = ['sex', 'age', 'address', 'studytime', 'failures', 'absences', 'G1', 'G2']
X = pd.get_dummies(df[features], columns=['sex', 'address'])  # Convertendo categóricas
y = np.where(df['G3'] >= 10, 1, 0)

# Normalização
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Criar a variável alvo: 1 para G3 >= 10 (aprovado), 0 caso contrário
df['aprovado'] = df['G3'].apply(lambda x: 1 if x >= 10 else 0)

# Divisão treino-teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

3.5– Verificação e correção de desbalanceamento:

```
# Verificar o balanceamento inicial
print("Balanceamento antes do oversampling:")
print(df['aprovado'].value_counts(normalize=True))

# Aplicar oversampling
ros = RandomOverSampler()
X_resampled, y_resampled = ros.fit_resample(X, y)
```

3.6– Treinamento do modelo MLP:

```
# Criando o modelo
modelo = MLPClassifier(
    hidden_layer_sizes=(1000, 1000),  # duas camadas com 1000 neurônios cada
    activation='relu',
    max_iter=1000,
    random_state=42
)

# Treinando o modelo
modelo.fit(X_train, y_train)
```

3.7– Avaliação do modelo:

```
y_pred = modelo.predict(X_test)

print("Relatório de Classificação:")
print(classification_report(y_test, y_pred, target_names=['Reprovado', 'Aprovado']))

print("\nMatriz de Confusão:")
print(confusion_matrix(y_test, y_pred))

acuracia = accuracy_score(y_test, y_pred)
print(f"\n\nAcurácia do modelo: {acuracia:.2%}")
```

3.8 - Interface Interativa:

```
print("\nSimulador de Aprovação Escolar")
print("Informe os dados do aluno:")

sex = input("Sexo (M/F): ").upper()
age = int(input("Idade: "))
address = input("Endereço (U=Urbano, R=Rural): ").upper()
studytime = int(input("Tempo de estudo (1:<2h, 2:2-5h, 3:5-10h, 4:>10h): "))
failures = int(input("Número de reprovações anteriores: "))
absences = int(input("Número de faltas: "))
G1 = int(input("Nota do primeiro período (0-20): "))
G2 = int(input("Nota do segundo período (0-20): "))

# Exibir os dados informados pelo usuário
print("\nDados informados:")
print(f"- Sexo: {sex}")
print(f"- Idade: {age} anos")
print(f"- Endereço: {'Urbano' if address == 'U' else 'Rural'}")
print(f"- Tempo de estudo: {studytime} ({'<2h' if studytime == 1 else '2-5h' if studytime == 2 else '5-10h' if studytime == 3 else '>10h'})")
print(f"- Reprovações anteriores: {failures}")
print(f"- Faltas: {absences}")
print(f"- Nota G1: {G1}")
print(f"- Nota G2: {G2}")

# Preparar os dados de entrada
input_data = pd.DataFrame({
    'age': [age],
    'studytime': [studytime],
    'failures': [failures],
    'absences': [absences],
    'G1': [G1],
    'G2': [G2],
    'sex_F': [1 if sex == 'F' else 0],
    'sex_M': [1 if sex == 'M' else 0],
    'address_R': [1 if address == 'R' else 0],
    'address_U': [1 if address == 'U' else 0]
})

# Garantir mesma ordem de colunas
input_data = input_data[X_train.columns] if hasattr(modelo, 'feature_names_in_') else input_data

# Normalizar e prever
input_data = scaler.transform(input_data)
predicao = modelo.predict(input_data)[0]
prob = modelo.predict_proba(input_data)[0][predicao]

print(f"\nResultado: {'Aprovado' if predicao == 1 else 'Reprovado'} (Probabilidade: {prob:.1%})")
```

4. CONSIDERAÇÕES FINAIS:

Este trabalho teve como objetivo investigar a eficácia das Redes Neurais Artificiais (RNA), especificamente o Perceptron Multicamadas (MLP), na predição do desempenho escolar de alunos do ensino secundário. A partir da aplicação do modelo sobre o conjunto de dados student-por.csv, foi possível obter uma acurácia superior a 92%, evidenciando o potencial dessa abordagem para identificar estudantes em risco de reprovação.

A metodologia aplicada foi cuidadosamente estruturada, envolvendo desde a análise exploratória inicial até o treinamento e avaliação do modelo. Técnicas como o pré-processamento de variáveis categóricas, padronização com StandardScaler e correção de desbalanceamento com RandomOverSampler contribuíram para melhorar a qualidade do treinamento.

A construção de uma interface de linha de comando tornou o modelo mais acessível a usuários não técnicos, permitindo simular cenários e interpretar os resultados de forma prática e objetiva.

Como trabalho futuro, recomenda-se explorar outras arquiteturas de RNA, como redes convolucionais (CNN) para análise de dados temporais e híbridos, além da integração com sistemas escolares via uma interface gráfica interativa ou aplicação web, visando uso em tempo real. A incorporação de novos atributos, como desempenho em avaliações externas ou participação em atividades extracurriculares, também pode enriquecer os modelos de predição.

5. REFERÊNCIAS BIBLIOGRÁFICAS:

CORTEZ, Paulo; SILVA, Alice. Using Data Mining to Predict Secondary School Student Performance. In: Proceedings of 5th Future Business Technology Conference (FUBUTEC 2008), Porto, Portugal, 2008. Disponível em: https://archive.ics.uci.edu/ml/datasets/Student+Performance. Acesso em: 24 maio 2025.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, v. 12, p. 2825–2830, 2011. Disponível em: https://scikit-learn.org. Acesso em: 24 maio 2025.

VAN DER WALT, S.; COLBERT, S. C.; VAROQUAUX, G. The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering, v. 13, n. 2, p. 22–30, 2011.

HUNTER, J. D. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, v. 9, n. 3, p. 90–95, 2007.

WASKOM, M. L. Seaborn: Statistical Data Visualization. Journal of Open Source Software, v. 6, n. 60, p. 3021, 2021. Disponível em: https://seaborn.pydata.org. Acesso em: 24 maio 2025.

LEMAITRE, G.; NAITO, K.; ARIAS, J.; KOVALEVSKI, A. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research, 18(17), 1–5, 2017. Disponível em: https://imbalanced-learn.org. Acesso em: 24 maio 2025.

Python Software Foundation. Python Language Reference, version 3.10. Disponível em: https://www.python.org. Acesso em: 24 maio 2025.