

✓ Curso: Redes Neurais e Deep Learning

Prof. Denilson Alves Pereira <https://sites.google.com/ufla.br/denilsonpereira/> Departamento de Ciência da Computação - Instituto de Ciências Exatas e Tecnológicas - Universidade Federal de Lavras

Atividade Prática 03

Tempo estimado para execução: 3 horas

Versão: Junho, 2021

✓ Projeto Final

O objetivo da atividade é desenvolver um projeto prático livre utilizando o conhecimento adquirido no curso. Você deve escolher um *dataset* para um problema de classificação, ler e efetuar o pré-processamento desse conjunto de dados e configurar uma rede neural para efetuar a classificação. Execute as seguintes etapas: definição, compilação e treinamento do modelo, avaliação e predição no conjunto de teste. Você deve avaliar diversas configurações para a sua rede neural, de forma a obter um resultado satisfatório. Verifique na literatura os melhores resultados obtidos com o *dataset*, se você estiver usando um *dataset* público.

Importante:

Documente cada etapa do seu código. Crie células de *Markdown* com textos explicativos e links para referências. Adicione comentários ao seu código.

DATASET

O dataset escolhido foi de Student Performance Factors, você pode baixá-lo no link a seguir: <https://www.kaggle.com/datasets/lainguyn123/student-performance-factors/data>. Também é possível importá-lo diretamente do kaggle como eu fiz.

PROBLEMA ESCOLHIDO E MODELO

O modelo desenvolvido tem como objetivo classificar o desempenho dos alunos em três categorias: abaixo da média, média e acima da média, com base em vários fatores que influenciam o desempenho acadêmico. Esses fatores foram extraídos de um conjunto de dados que contém informações relevantes sobre os alunos.

Após a construção e treinamento do modelo de rede neural, os resultados de avaliação indicaram uma acurácia de 92,08%, o que demonstra uma performance robusta. Isso significa que o modelo foi capaz de prever corretamente 1826 dos 1983 exemplos testados, resultando em uma taxa de acerto bastante satisfatória.

Detalhes dos Resultados: Loss: 0.1860 - Um valor relativamente baixo, indicando que o modelo está se ajustando bem aos dados de treinamento. Acurácia: 92,08% - Isso reflete a capacidade do modelo de realizar previsões precisas em novos dados. Esses resultados sugerem que o modelo pode ser utilizado efetivamente para prever o desempenho de alunos com base nos fatores considerados, oferecendo insights valiosos que podem ser usados por educadores e instituições de ensino para identificar alunos que possam precisar de suporte adicional ou para estratégias de ensino personalizadas.

Em resumo, a aplicação deste modelo pode contribuir significativamente para a melhoria do desempenho acadêmico, ajudando a identificar áreas de intervenção e apoio de forma proativa.

✓ PACOTES

```
### INICIE O CÓDIGO AQUI ### (várias linhas de código / várias células)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow import keras
from keras import optimizers
from keras.optimizers import Adam
import os
```

definindo o caminho para o diretório do conjunto de dados:

```
student_performance_factors_path = '/root/.cache/kagglehub/datasets/lainguyn123/student-performance-factors/versions/6'
```


Clique duas vezes (ou pressione "Enter") para editar

✓ IMPORTAÇÃO DO DATASET DIRETAMENTE DO *KAGGLE*

```
# Listando todos os arquivos no diretório do conjunto de dados
for dirname, _, filenames in os.walk(student_performance_factors_path):
    for filename in filenames:
        print(os.path.join(dirname, filename))

student_performance_csv_path = os.path.join(student_performance_factors_path, "StudentPerformanceFactors.csv")


print("Caminho do arquivo CSV:", student_performance_csv_path)
```



```
/root/.cache/kagglehub/datasets/lainguy123/student-performance-factors/versions/6/StudentPerformanceFactors.csv
Caminho do arquivo CSV: /root/.cache/kagglehub/datasets/lainguy123/student-performance-factors/versions/6/StudentPerformanceFactors.csv
```

✓ PRÉ PROCESSAMENTO DE DADOS

```
df = pd.read_csv(student_performance_csv_path)
print("\nPrimeiras 5 instâncias:")
print(df.head())
```



```
Primeiras 5 instâncias:
Hours_Studied  Attendance  Parental_Involvement  Access_to_Resources  \
0              23          84                    Low              High
1              19          64                    Low             Medium
2              24          98                  Medium             Medium
3              29          89                    Low             Medium
4              19          92                  Medium             Medium

Extracurricular_Activities  Sleep_Hours  Previous_Scores  Motivation_Level  \
0                        No              7              73              Low
1                        No              8              59              Low
2                       Yes              7              91             Medium
3                       Yes              8              98             Medium
4                       Yes              6              65             Medium


Internet_Access  Tutoring_Sessions  Family_Income  Teacher_Quality  \
0             Yes                  0             Low             Medium
1             Yes                  2             Medium            Medium
2             Yes                  2             Medium            Medium
3             Yes                  1             Medium            Medium
4             Yes                  3             Medium             High

School_Type  Peer_Influence  Physical_Activity  Learning_Disabilities  \
0      Public      Positive              3              No
1      Public      Negative              4              No
2      Public      Neutral              4              No
3      Public      Negative              4              No
4      Public      Neutral              4              No

Parental_Education_Level  Distance_from_Home  Gender  Exam_Score
0             High School              Near  Male           67
1              College              Moderate  Female          61
2      Postgraduate              Near  Male           74
3             High School              Moderate  Male           71
4              College              Near  Female          70
```

Informações sobre os dados:

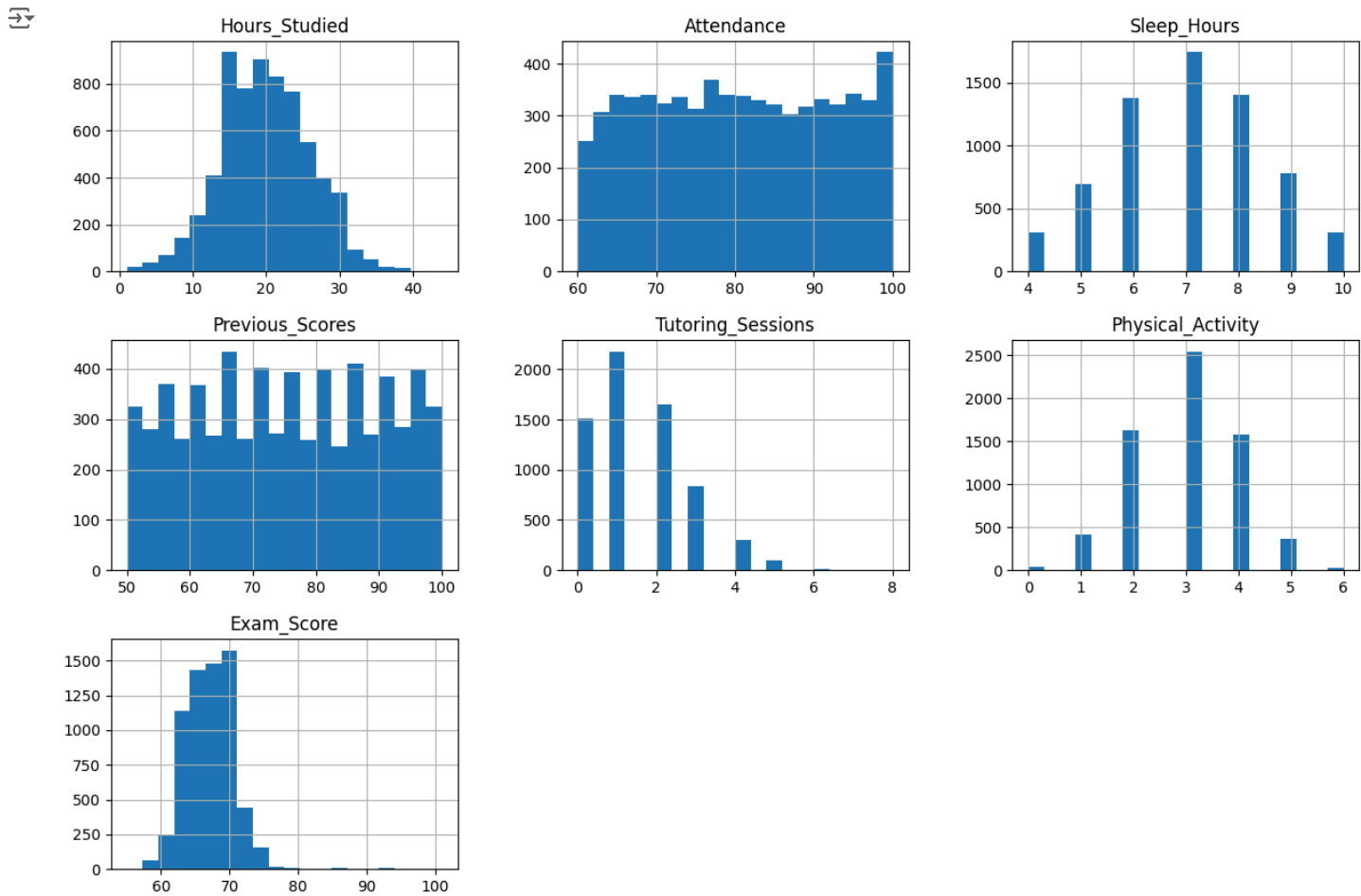
```
print("\nInformações sobre os dados:")
df.info()
```



```
Informações sobre os dados:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6607 entries, 0 to 6606
Data columns (total 20 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Hours_Studied                        6607 non-null  int64
1   Attendance                          6607 non-null  int64
2   Parental_Involvement                6607 non-null  object
3   Access_to_Resources                 6607 non-null  object
4   Extracurricular_Activities          6607 non-null  object
5   Sleep_Hours                        6607 non-null  int64
6   Previous_Scores                    6607 non-null  int64
7   Motivation_Level                    6607 non-null  object
8   Internet_Access                     6607 non-null  object
9   Tutoring_Sessions                  6607 non-null  int64
10  Family_Income                       6607 non-null  object
11  Teacher_Quality                     6529 non-null  object
12  School_Type                         6607 non-null  object
13  Peer_Influence                      6607 non-null  object
14  Physical_Activity                   6607 non-null  int64
15  Learning_Disabilities               6607 non-null  object
16  Parental_Education_Level            6517 non-null  object
17  Distance_from_Home                  6540 non-null  object
18  Gender                             6607 non-null  object
19  Exam_Score                          6607 non-null  int64
dtypes: int64(7), object(13)
memory usage: 1.0+ MB
```

✓ Visualização da distribuição dos dados

```
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
df.hist(column=numeric_columns, bins=20, figsize=(15, 10))
plt.show()
```



```
print("\nNormalização dos dados...")
```

```
Normalização dos dados...
```

Convertendo variáveis categóricas em numéricas

```
# Convertendo variáveis categóricas em numéricas
le = preprocessing.LabelEncoder()
categorical_columns = df.select_dtypes(include=['object']).columns
for column in categorical_columns:
    df[column] = le.fit_transform(df[column])
    print(f"Coluna {column} convertida para numérica")

print("\nPrimeiras 5 instâncias após normalização:")
print(df.head())
```

Primeiras 5 instâncias após normalização:

	Hours_Studied	Attendance	Parental_Involvement	Access_to_Resources	\
0	23	84	1	0	
1	19	64	1	2	
2	24	98	2	2	
3	29	89	1	2	
4	19	92	2	2	

	Extracurricular_Activities	Sleep_Hours	Previous_Scores	Motivation_Level	\
0	0	7	73	1	
1	0	8	59	1	
2	1	7	91	2	
3	1	8	98	2	
4	1	6	65	2	

	Internet_Access	Tutoring_Sessions	Family_Income	Teacher_Quality	\
0	1	0	1	2	
1	1	2	2	2	
2	1	2	2	2	
3	1	1	2	2	
4	1	3	2	0	

	School_Type	Peer_Influence	Physical_Activity	Learning_Disabilities	\
0	1	2	3	0	
1	1	0	4	0	
2	1	1	4	0	
3	1	0	4	0	
4	1	1	4	0	

	Parental_Education_Level	Distance_from_Home	Gender	Exam_Score
0	1	2	1	67

1	0	1	0	61
2	2	2	1	74
3	1	1	1	71
4	0	2	0	70


✖ Categorização da Performance

```
def categorize_performance(value):
    if value < 70:
        return 0 # Abaixo da média
    elif 70 <= value < 85:
        return 1 # Média
    else:
        return 2 # Acima da média

# Aplicando a categorização
df['Exam_Score'] = df['Exam_Score'].apply(categorize_performance)
```

Verificando os valores únicos

```
print("\nDistribuição das classes após categorização:")
print(df['Exam_Score'].value_counts())
```

 Distribuição das classes após categorização:  
Exam\_Score  
0 4982  
1 1590  
2 35  
Name: count, dtype: int64

✖ Dividindo dados entre features e target

```
X = df.drop('Exam_Score', axis=1)
Y = df['Exam_Score']
```

✖ Dividindo o código entre teste e treinamento (70% será destinado ao treinamento)


```
train_set_X, test_set_X, train_set_Y, test_set_Y = train_test_split(X, Y, test_size=0.30, random_state=42)
```

✖ Normalização dos dados usando Min-Max

```
print("\nNormalização Min-Max dos dados...")
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(train_set_X)
X_test_scaled = scaler.transform(test_set_X)

n = train_set_X.shape[1] # número de atributos
m = train_set_X.shape[0] # número de exemplos de treinamento

print("\nNúmero de atributos: n = " + str(n))
print("Número de exemplos de treinamento: m = " + str(m))
print("Formato do conjunto de treino X: " + str(train_set_X.shape))
print("Formato do conjunto de treino Y: " + str(train_set_Y.shape))
print("Formato do conjunto de teste X: " + str(test_set_X.shape))
print("Formato do conjunto de teste Y: " + str(test_set_Y.shape))
```

 Normalização Min-Max dos dados...

Número de atributos: n = 19  
Número de exemplos de treinamento: m = 4624  
Formato do conjunto de treino X: (4624, 19)  
Formato do conjunto de treino Y: (4624,)  
Formato do conjunto de teste X: (1983, 19)  
Formato do conjunto de teste Y: (1983,)

✖ Configuração do modelo

```
print("\nDefinindo a estrutura da rede...")

inputs = keras.Input(shape=(train_set_X.shape[1],))

layer1 = keras.layers.Dense(units=128, activation='relu')(inputs)
dropout1 = keras.layers.Dropout(0.3)(layer1)

layer2 = keras.layers.Dense(units=64, activation='relu')(dropout1)
dropout2 = keras.layers.Dropout(0.2)(layer2)

layer3 = keras.layers.Dense(units=32, activation='relu')(dropout2)
dropout3 = keras.layers.Dropout(0.2)(layer3)
```

```
outputs = keras.layers.Dense(units=3, activation="softmax")(dropout3)

model = keras.models.Model(inputs=inputs, outputs=outputs)

print("\nEstrutura da rede:")
model.summary()
```

Definindo a estrutura da rede...

Estrutura da rede:  
Model: "functional\_2"

Layer (type)	Output Shape	Param #
input_layer_2 ( <a href="#">InputLayer</a> )	(None, 19)	0
dense_8 ( <a href="#">Dense</a> )	(None, 128)	2,560
dropout_6 ( <a href="#">Dropout</a> )	(None, 128)	0
dense_9 ( <a href="#">Dense</a> )	(None, 64)	8,256
dropout_7 ( <a href="#">Dropout</a> )	(None, 64)	0
dense_10 ( <a href="#">Dense</a> )	(None, 32)	2,080
dropout_8 ( <a href="#">Dropout</a> )	(None, 32)	0
dense_11 ( <a href="#">Dense</a> )	(None, 3)	99

Total params: 12,995 (50.76 KB)  
Trainable params: 12,995 (50.76 KB)  
Non-trainable params: 0 (0.00 KB)

Compilação do modelo

```
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
print("Valores únicos em train_set_Y:", np.unique(train_set_Y))
```

Valores únicos em train\_set\_Y: [0 1 2]

Treinamento do modelo

```
print("\nTreinando o modelo...")
history = model.fit(X_train_scaled, train_set_Y,
                   batch_size=32,
                   epochs=50,
                   validation_split=0.2)
```


Epoch 22/50

```
116/116 ----- 1s 3ms/step - accuracy: 0.9290 - loss: 0.1815 - val_accuracy: 0.9081 - val_loss: 0.2960
Epoch 45/50
116/116 ----- 0s 2ms/step - accuracy: 0.9336 - loss: 0.1700 - val_accuracy: 0.9146 - val_loss: 0.2908
Epoch 46/50
116/116 ----- 0s 3ms/step - accuracy: 0.9427 - loss: 0.1610 - val_accuracy: 0.9114 - val_loss: 0.2870
Epoch 47/50
116/116 ----- 1s 3ms/step - accuracy: 0.9328 - loss: 0.1807 - val_accuracy: 0.9114 - val_loss: 0.2850
Epoch 48/50
116/116 ----- 1s 4ms/step - accuracy: 0.9455 - loss: 0.1531 - val_accuracy: 0.9114 - val_loss: 0.3268
Epoch 49/50
116/116 ----- 1s 4ms/step - accuracy: 0.9356 - loss: 0.1650 - val_accuracy: 0.9146 - val_loss: 0.3044
Epoch 50/50
116/116 ----- 1s 4ms/step - accuracy: 0.9323 - loss: 0.1700 - val_accuracy: 0.9005 - val_loss: 0.3206
```

```
# Treinamento do modelo
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

▼ Avaliação do modelo

```
print("\nAvaliando o modelo...")
loss, accuracy = model.evaluate(X_test_scaled, test_set_Y)
print(f"\nLoss: {loss:.2f}")
print(f"Accuracy: {accuracy:.2f}")
```

 Avaliando o modelo...


62/62 ----- 0s 2ms/step - accuracy: 0.9270 - loss: 0.1860

Loss: 0.22  
Accuracy: 0.92

▼ Predição

```
print("\nRealizando predições...")
predictions = model.predict(X_test_scaled)
predicted_classes = np.argmax(predictions, axis=1)
correct_labels = test_set_Y.values

num_correct = sum(predicted_classes == correct_labels)
print("\nNúmero de acertos:", num_correct)
print("Total de exemplos:", len(correct_labels))
print(f"Acurácia: {(num_correct/len(correct_labels))*100:.2f}%")
```

 Realizando predições...

62/62 ----- 1s 6ms/step

Número de acertos: 1826  
Total de exemplos: 1983  
Acurácia: 92.08%

### TERMINE O CÓDIGO AQUI ###