REVISTA ERR01 ISSN: 2595-1378

AN ANALYSIS OF RANDOM FOREST MACHINE LEARNING MODELS AND ARTIFICIAL NEURAL NETWORKS

UMA ANÁLISE DE MODELOS DE APRENDIZAGEM DE MÁQUINA DE FLORESTA ALEATÓRIA E REDES NEURAIS ARTIFICIAIS

UN ANÁLISIS DE MODELOS DE APRENDIZAJE AUTOMÁTICO DE BOSQUE ALEATORIO Y REDES NEURONALES ARTIFICIALES

10.56238/ERR01v10n2-004

Alfredo Nazareno Pereira Boente[1], Renata Miranda Pires Boente[2] and Marianne Luana Bueno[3]

ABSTRACT
This academic work developed an analysis using Artificial Intelligence, through machine learning techniques, with the objective of investigating how a student's social context can impact their score on the Mathematics and its Technologies test of ENEM. For this, open microdata related to the 2022 edition of ENEM were used, which include social, educational, and economic variables of the participants. These data were organized and processed with the help of machine learning algorithms, specifically Random Forest and Artificial Neural Network models, to compare the performance of each technique based on criteria such as accuracy, robustness, and interpretability. At the end of the research, the model that presented the best performance in identifying the correlation between the analyzed factors was highlighted, offering support for understanding the most influential variables and promoting a critical reflection on the role of the government in formulating public policies aimed at improving education in the country.

Keywords: Machine Learning. Random Florest. Artificial Neural Networks.

RESUMO
Este trabalho acadêmico desenvolveu uma análise utilizando Inteligência Artificial, por meio de técnicas de aprendizado de máquina, com o objetivo de investigar como o contexto social de um aluno pode impactar sua nota na prova de Matemática e Tecnologias do ENEM. Para isso, foram utilizados microdados abertos referentes à edição de 2022 do

[1]Dr. in Production Engineering
Federal University of Rio de Janeiro, COPPE/UFRJ
boente@nce.ufrj.br
https://orcid.org/0000-0002-2718-4917
[2]Doctorate student in History of Science and Technology and Epistemology
Federal University of Rio de Janeiro, HCTE/UFRJ
renata@hcte.ufrj.br
https://orcid.org/0000-0001-7856-5691
[3]Computer Engineer
Veiga de Almeida University, UVA
mariannebueno@live.com
https://orcid.org/0009-0005-4485-8124

ENEM, que incluem variáveis sociais, educacionais e econômicas dos participantes. Esses dados foram organizados e processados com o auxílio de algoritmos de aprendizado de máquina, especificamente modelos de Floresta Aleatória e Redes Neurais Artificiais, para comparar o desempenho de cada técnica com base em critérios como acurácia, robustez e interpretabilidade. Ao final da pesquisa, foi destacado o modelo que apresentou melhor desempenho na identificação da correlação entre os fatores analisados, oferecendo subsídios para a compreensão das variáveis mais influentes e promovendo uma reflexão crítica sobre o papel do governo na formulação de políticas públicas voltadas à melhoria da educação no país.

Palavras-chave: Aprendizado de Máquina. Floresta Aleatória. Redes Neurais Artificiais.

RESUMEN

Este trabajo académico desarrolló un análisis mediante Inteligencia Artificial, mediante técnicas de aprendizaje automático, con el objetivo de investigar cómo el contexto social de un estudiante puede influir en su puntuación en la prueba de Matemáticas y Tecnologías del ENEM. Para ello, se utilizaron microdatos abiertos de la edición 2022 del ENEM, que incluyen variables sociales, educativas y económicas de los participantes. Estos datos se organizaron y procesaron con la ayuda de algoritmos de aprendizaje automático, específicamente modelos de Bosque Aleatorio y Redes Neuronales Artificiales, para comparar el rendimiento de cada técnica con base en criterios como precisión, robustez e interpretabilidad. Al final de la investigación, se destacó el modelo con el mejor rendimiento en la identificación de la correlación entre los factores analizados, lo que ofrece apoyo para comprender las variables más influyentes y promueve una reflexión crítica sobre el papel del gobierno en la formulación de políticas públicas destinadas a mejorar la educación en el país.

Palabras clave: Aprendizaje automático. Bosque aleatorio. Redes neuronales artificiales.

INTRODUCTION

The National High School Exam (ENEM) has established itself as the main access route to higher education in Brazil. The affirmative action policies implemented by the government, integrated with programs that use ENEM scores to fill vacancies, have contributed to increasing social diversity in public and private higher education institutions. However, despite these advances in democratizing access, significant inequalities persist in student performance, especially in Mathematics and its Technologies, reflecting the different social contexts in which they are inserted.

According to the 2022 Higher Education Census, conducted by the National Institute of Studies and Educational Research Anísio Teixeira (INEP), student enrollment in federal universities through affirmative action increased by 167% in the last decade. However, in that same year, Brazil was among the 20 countries with the worst performance in mathematics, according to data from PISA (Programme for International Student Assessment), with an average of 379 points, well below the average of 479 recorded by OECD (Organization for Economic Cooperation and Development) countries.

These data reveal that, although access to universities has become broader, Brazilian basic education still faces serious challenges in terms of providing quality education. These challenges are often related to broader social factors. A survey by the “Todos Pela Educação” movement, based on IBGE data between 2012 and 2022, showed that only recently did the proportion of young black people with access to education at the appropriate age match that of young white people a decade earlier.

Given this scenario, it is possible to understand that multiple factors directly influence students' preparation for the ENEM, considering the disparities in the environments in which they live, the opportunities they receive, and the social experiences they accumulate throughout their lives. Social inequality, therefore, is significantly reflected in the performance obtained in the exam. Therefore, scientific research capable of identifying which elements impact students' proficiency in the test becomes essential, with the aim of proposing more effective strategies for promoting educational equity in the country.

METODOLOGY

This study adopts a qualitative approach with an exploratory and experimental nature, based on a literature review and document analysis, with the objective of critically examining the algorithms that structure machine learning models, especially Random Forest and Artificial Neural Networks. The research aims to contextualize these models within the training process based on public microdata from ENEM, seeking to understand how the algorithms operate and what behavior patterns they present during data analysis.

The investigation promotes a comparison between the two learning models, considering performance metrics, such as accuracy and stability in predictions. Based on the results obtained, the aim is to identify which of the algorithms demonstrated greater effectiveness in processing and interpreting the data, allowing more reliable inferences about the factors that influence students' results in the test. In addition, the choice for this type of methodology is justified by the need to empirically explore the functionalities and limitations of the applied models, which also gives the work an experimental nature.

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

In the 1940s, researchers Warren McCulloch, a neurophysiologist, and Walter Pitts, a mathematician, began a pioneering study on the functioning of biological neurons in the human brain. Motivated by the intention of translating this neurological process into the computational universe, they developed the first artificial model of a neuron, presented in the article “A Logical Calculus of the Ideas Immanent in Nervous Activity” (MCCULLOCH and PITTS, 1943). This work, even before the formal definition of the concept of artificial intelligence, is widely recognized as the origin of artificial neural networks and one of the first attempts to create a machine capable of simulating human cognitive operations. Their proposal became a milestone in the history of computing, strongly influencing future research in Artificial Intelligence (AI) and the development of intelligent systems (LIMA, 2024).

According to Taulli (2020), artificial intelligence is a branch of computer science focused on the development of machines and algorithms that seek to emulate the human ability to think, decide and perform tasks, whether routine or highly complex. AI therefore aims to provide computer systems with autonomy and the ability to adapt to different contexts, bringing them closer to intelligent human behavior.

One of the central fields within AI is machine learning, which is based on mathematical models applied to data sets with the aim of making predictions, inferences and classifications. According to Breiman (2001), machine learning models can be organized into two broad categories: supervised and unsupervised. Supervised models operate with labeled data, that is, already categorized, while unsupervised models analyze unstructured data, identifying patterns autonomously. These approaches have been widely used in modern applications, from recommendation systems to cybersecurity mechanisms and medical diagnostics, highlighting the importance and versatility of machine learning in the current digital age.

ALGORITHM TRAINING MODELS

Machine learning algorithm training models can be classified as supervised and unsupervised. In supervised models, the algorithm is trained using a data set consisting of pairs consisting of one or more input variables and one output variable (output or target). We call these labeled data because the expected outputs for the inputs are known in advance. Thus, when training a model, the output data serves as “supervisors” to compare the predicted data and guide the model through error measures between the predictions and the true labels, such as the loss function.

In unsupervised models, the algorithm is trained with unlabeled data, that is, in sets where there is no output variable as a reference. In this scenario, the model structures the data and applies predictions by discovering patterns between the input variables using techniques that do not depend on guidance, such as clustering or dimensionality reduction (BREIMAN, 2001).

In this context, regarding machine learning models, those that use classification algorithms and regression algorithms stand out (LIAW and WIENER, 2002).

Classification algorithms have categorical target variables, through mathematical probability calculations for each class. In this format, the models learn from the data to classify an observation based on the characteristics (input variables) presented. For example, a bank can analyze, through the customer's history, whether they are eligible to take out a loan.

On the other hand, algorithms that have continuous target variables (predicted values over which there is no control), such as the value of the loan made available by the bank to the customer, are called regression algorithms.

ENEM OPEN DATA

Since 2009, INEP has made data from the ENEM publicly available, with the aim of strengthening transparency and facilitating access to educational information. Based on this initiative, this study used microdata from the 2022 edition of the ENEM to apply machine learning models to analyze the behavior of variables and their possible correlations (INEP, 2024).

The data set consists of 76 variables, classified into different categories: participant information, educational institution data, test location data, objective test results, essay performance, and responses to the socioeconomic questionnaire. To develop computational models, 17 variables were selected to serve as inputs for the Random Forest and Artificial Neural Network algorithms. The implementation of these models was carried out using the Python programming language, allowing the evaluation of significant patterns and relationships between the analyzed data.

RANDOM FOREST MACHINE LEARNING MODEL

Before we even talk about Random Forest, it is necessary to clarify the concept of decision tree, which refers to a technique that hierarchically outlines, in the form of a tree, a visual and logical representation of the trajectories of decisions made and their results. Its structure begins with a single node called the “root node”, which branches out, through actions or responses to the previous node, into possible results (NAPOLEÃO, 2024). These results can add other nodes to the structure, called “decision nodes”, which grow the tree depending on its level of complexity. In the end, we have the nodes resulting from the set of decisions made, known as “leaf nodes”. In the context of machine learning, the Decision Tree is a supervised predictive model proposed by Breiman (1996) known as CART (Classification and Regression Tree). The algorithm maps a set of inputs to an output through a series of decisions based on the characteristics of these inputs.

Random Forest is an algorithm developed by Leo Breiman to solve the overfitting limitation that Decision Trees have in their models. To achieve this, the algorithm uses the ensemble method, where several models operate together to obtain a single result. In the case of Random Forest, the models applied are Decision Trees (BREIMAN, 2001).

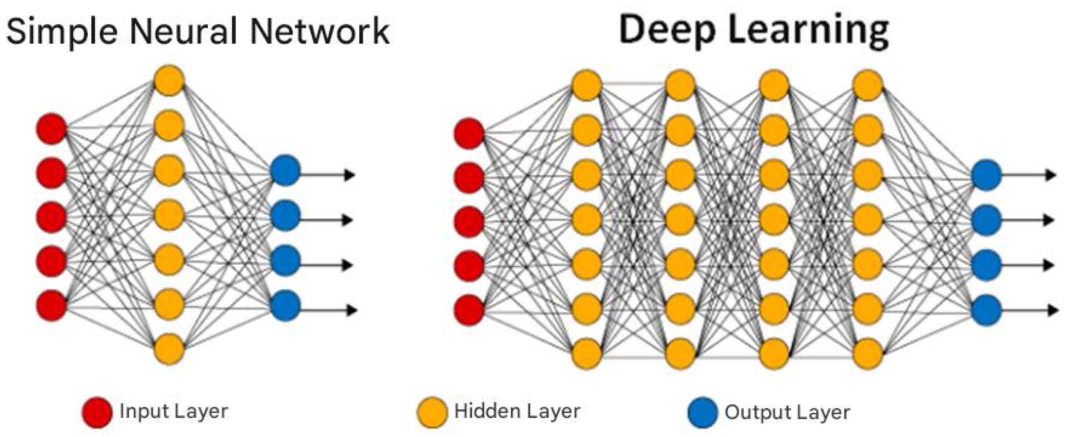
The Random Forest model presents a prediction based on the average of several predictive results generated by individual trees, in a reliable and accurate output.

ARTIFICIAL NEURAL NETWORKS MACHINE LEARNING MODEL

In 1958, researcher Frank Rosenblatt presented an innovative proposal: an artificial neural network algorithm inspired by the functioning of biological neurons in the human brain. This proposal resulted in the development of the “Perceptron”, considered one of the first machine learning models aimed at binary classification tasks with a single output. Although there were previous initiatives in the field of artificial intelligence, the Perceptron stood out for its well-structured mathematical basis, the possibilities for practical application and the way it expanded scientific interest in the area (ROSENBLATTEM, 1958). Although it had limitations in relation to solving non-linearly separable problems, its historical impact was significant, influencing generations of researchers and serving as a basis for the improvement of later models.

The algorithm's structure was designed based on a direct analogy with the components of a biological neuron. The model's inputs represent the dendrites, which are the branches responsible for receiving signals from other neurons. Each input is accompanied by a weight, which simulates the function of synapses, connections that regulate the transmission strength of nerve impulses. These weights determine the relative importance of each processed attribute. The bias plays the role of resting potential, being a fixed value that adjusts the activation of the artificial neuron, shifting the decision boundary. The axon, which in a biological neuron transmits the electrical signal to other neurons, in the Perceptron is represented by the activation function, which defines whether the neuron will be activated based on the total value obtained (BUENO and BOENTE, 2024). Thus, the Perceptron not only introduced fundamental concepts for the construction of neural network models but also provided a new understanding of the possibility of reproducing human cognitive processes in computer systems. Its legacy remains one of the pillars of modern artificial intelligence.

Fig. 1 - Comparison between simple neural network model and deep learning model



Source: DATA SCIENCE ACADEMY (2024).

According to Goodfellow, Bengio and Courville (2017), the difference between a simple neural network model and a deep learning model is the number of hidden layers implemented in its algorithm. While a simple neural network consists of only one input layer, one hidden layer and one output layer, a deep learning model can use two or more layers, as illustrated in Fig. 1.

METHODOLOGICAL PROCEDURES

To broaden the understanding of the topic addressed and contextualize it through relevant social data, this study used information provided by official institutions. To support the technical basis of the research, according to Bueno and Boente (2024), reference scientific articles in the field of Artificial Intelligence were consulted, as well as specialized works on machine learning models and their implementations in the Python language.

The research uses as its main source the open microdata of ENEM, specifically those related to the year 2022. The adopted approach includes a thorough exploratory analysis of the data, with the aim of extracting an initial understanding of its composition. This stage included the identification and categorization of the variables present, the examination of the data structure, the verification of the occurrence of missing or null values, the total number of available records and the detection of possible anomalies or inconsistencies.

This procedure is essential to ensure the quality of the data used in machine learning models, since it provides the necessary basis for the pre-processing and attribute selection stage. In addition, exploratory analysis contributes to a critical reading of educational data in Brazil, allowing the study to advance with greater precision in the application of the proposed computational models.

A sample was used with a database of all students who took the ENEM in 2022, which was representative of the universe studied, since only observations where none of their variables were empty or null and students with the high school status as “completed” or “to be completed” were considered.

The study presented a quantitative approach, since numerical and categorical variables of the students were analyzed where the research methods used included statistical and machine learning techniques to analyze the data in a comprehensive and detailed manner. This involved applying descriptive statistical methods to understand the distribution and characteristics of the variables, as well as using regression models and other quantitative analysis techniques to explore relationships between the variables and students' grades.

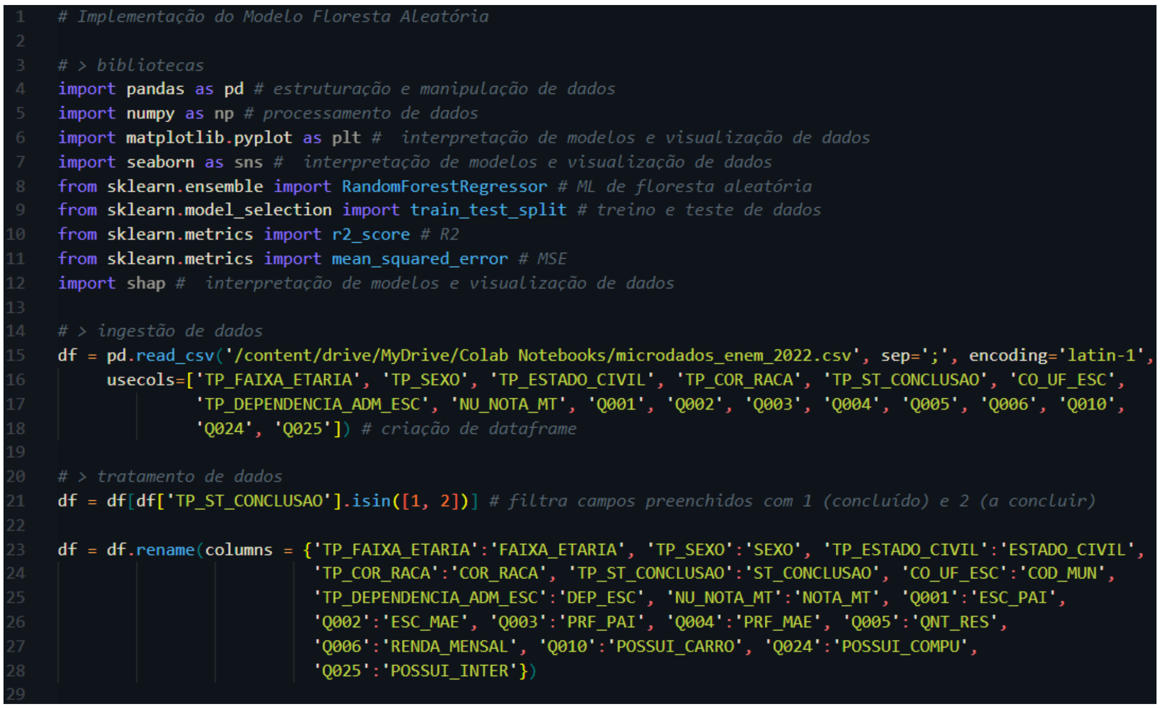
RESULTS AND DISCUSSIONS

The results of this study are presented, allowing discussions on the comparative analysis of machine learning models, Random Forest and Artificial Neural Network.

Random Florest Model

To apply the models presented in this study, Python was used, with the Google Colab development environment, whose implementation of the Random Forest model, its structuring, manipulation and visualization of data, as well as for the application of the machine learning model, the following libraries were used: Pandas, NumPy, Seaborn, Matplotlib and Scikit-learn.

Fig. 2 - Excerpt from the program that performs data ingestion and processing



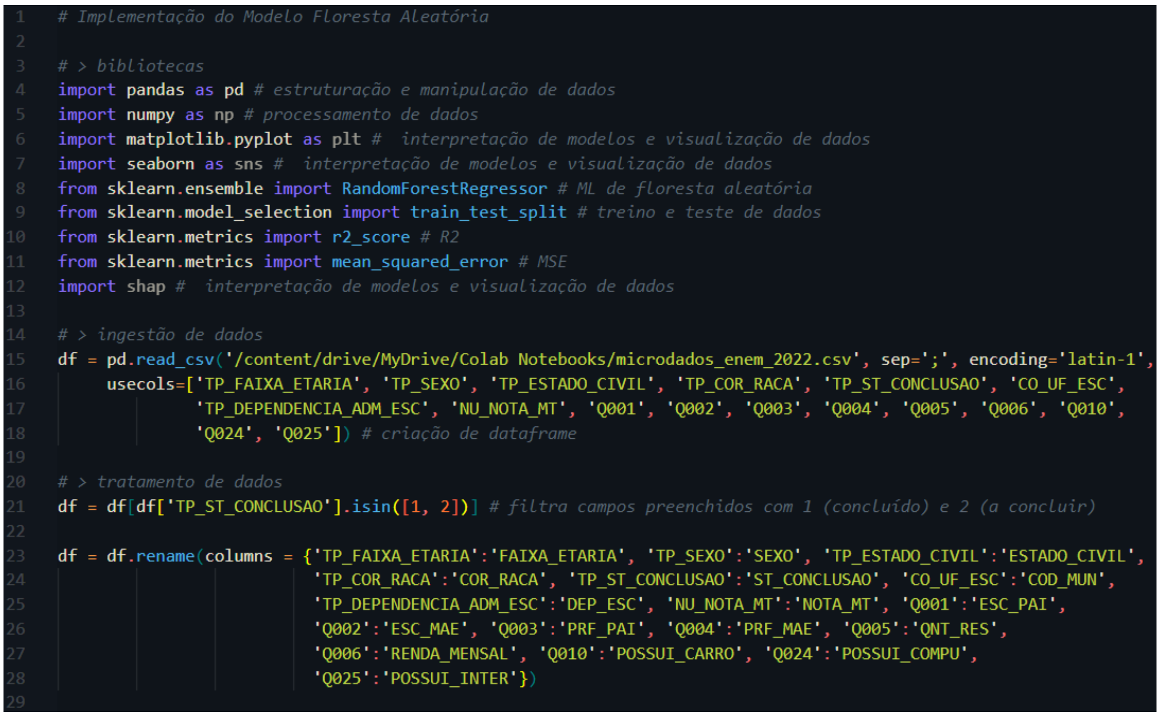
```python
# Implementação do Modelo Floresta Aleatória

# > bibliotecas
import pandas as pd # estruturação e manipulação de dados
import numpy as np # processamento de dados
import matplotlib.pyplot as plt #  interpretação de modelos e visualização de dados
import seaborn as sns #  interpretação de modelos e visualização de dados
from sklearn.ensemble import RandomForestRegressor # ML de floresta aleatória
from sklearn.model_selection import train_test_split # treino e teste de dados
from sklearn.metrics import r2_score # R2
from sklearn.metrics import mean_squared_error # MSE
import shap #  interpretação de modelos e visualização de dados

# > ingestão de dados
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/microdados_enem_2022.csv', sep=';', encoding='latin-1',
     usecols=['TP_FAIXA_ETARIA', 'TP_SEXO', 'TP_ESTADO_CIVIL', 'TP_COR_RACA', 'TP_ST_CONCLUSAO', 'CO_UF_ESC',
              'TP_DEPENDENCIA_ADM_ESC', 'NU_NOTA_MT', 'Q001', 'Q002', 'Q003', 'Q004', 'Q005', 'Q006', 'Q010',
              'Q024', 'Q025']) # criação de dataframe

# > tratamento de dados
df = df[df['TP_ST_CONCLUSAO'].isin([1, 2])] # filtra campos preenchidos com 1 (concluído) e 2 (a concluir)

df = df.rename(columns = {'TP_FAIXA_ETARIA':'FAIXA_ETARIA', 'TP_SEXO':'SEXO', 'TP_ESTADO_CIVIL':'ESTADO_CIVIL',
                          'TP_COR_RACA':'COR_RACA', 'TP_ST_CONCLUSAO':'ST_CONCLUSAO', 'CO_UF_ESC':'COD_MUN',
                          'TP_DEPENDENCIA_ADM_ESC':'DEP_ESC', 'NU_NOTA_MT':'NOTA_MT', 'Q001':'ESC_PAI',
                          'Q002':'ESC_MAE', 'Q003':'PRF_PAI', 'Q004':'PRF_MAE', 'Q005':'QNT_RES',
                          'Q006':'RENDA_MENSAL', 'Q010':'POSSUI_CARRO', 'Q024':'POSSUI_COMPU',
                          'Q025':'POSSUI_INTER'})
```

Source: Bueno and Boente (2024).

Fig. 2 illustrates the data ingestion process using the pd.read_csv() function from the Pandas library. The first argument is given by the Google Drive path to load the data from the file 'microdados_enem_2022.csv'. Next, we set the sep parameter to ';' to indicate that the file is separated by semicolons. The encoding parameter is specified as 'latin-1' to ensure that special characters are read correctly. Finally, we use the usecols parameter to select only the columns that are relevant to our study. The result of the function is assigned to the variable 'df'.

We then perform data processing to ensure its consistency and suitability for the proposed model. In this step, we apply filters and rename columns to facilitate understanding and manipulation of the data. After applying the filter, we rename the dataframe columns using the rename() method, assigning more descriptive and understandable names to the variables.

Fig. 3 - Excerpt from the program that handles missing values



```python
# Implementação do Modelo Floresta Aleatória

# > bibliotecas
import pandas as pd # estruturação e manipulação de dados
import numpy as np # processamento de dados
import matplotlib.pyplot as plt #  interpretação de modelos e visualização de dados
import seaborn as sns #  interpretação de modelos e visualização de dados
from sklearn.ensemble import RandomForestRegressor # ML de floresta aleatória
from sklearn.model_selection import train_test_split # treino e teste de dados
from sklearn.metrics import r2_score # R2
from sklearn.metrics import mean_squared_error # MSE
import shap #  interpretação de modelos e visualização de dados

# > ingestão de dados
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/microdados_enem_2022.csv', sep=';', encoding='latin-1',
     usecols=['TP_FAIXA_ETARIA', 'TP_SEXO', 'TP_ESTADO_CIVIL', 'TP_COR_RACA', 'TP_ST_CONCLUSAO', 'CO_UF_ESC',
              'TP_DEPENDENCIA_ADM_ESC', 'NU_NOTA_MT', 'Q001', 'Q002', 'Q003', 'Q004', 'Q005', 'Q006', 'Q010',
              'Q024', 'Q025']) # criação de dataframe

# > tratamento de dados
df = df[df['TP_ST_CONCLUSAO'].isin([1, 2])] # filtra campos preenchidos com 1 (concluído) e 2 (a concluir)

df = df.rename(columns = {'TP_FAIXA_ETARIA':'FAIXA_ETARIA', 'TP_SEXO':'SEXO', 'TP_ESTADO_CIVIL':'ESTADO_CIVIL',
                          'TP_COR_RACA':'COR_RACA', 'TP_ST_CONCLUSAO':'ST_CONCLUSAO', 'CO_UF_ESC':'COD_MUN',
                          'TP_DEPENDENCIA_ADM_ESC':'DEP_ESC', 'NU_NOTA_MT':'NOTA_MT', 'Q001':'ESC_PAI',
                          'Q002':'ESC_MAE', 'Q003':'PRF_PAI', 'Q004':'PRF_MAE', 'Q005':'QNT_RES',
                          'Q006':'RENDA_MENSAL', 'Q010':'POSSUI_CARRO', 'Q024':'POSSUI_COMPU',
                          'Q025':'POSSUI_INTER'})
```

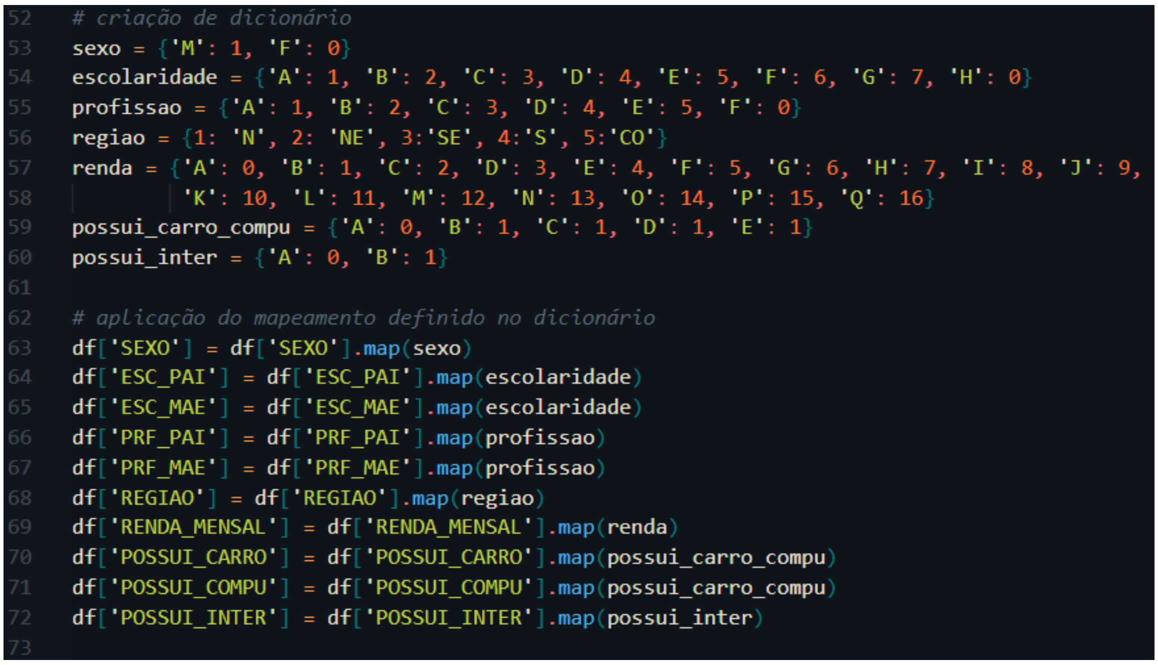
Source: Bueno and Boente (2024).

Next, we apply a treatment to deal with missing values using the dropna() function, removing records that have null values in specific columns. Fig. 3 illustres the handles missing values.

To facilitate data manipulation and analysis, we mapped some categorical variables to numerical values. We created variables corresponding to the characteristics and defined mapping dictionaries for each of them. From then on, we applied these mappings to the respective columns of the dataset using the map() method, as illustrated in Fig. 4.

Next, we use the pd.get_dummies() function from the Pandas library to perform One-Hot Encoding of the selected categorical variables.

Fig. 4 - Excerpt from the program that performs the mappings defined in the dictionary

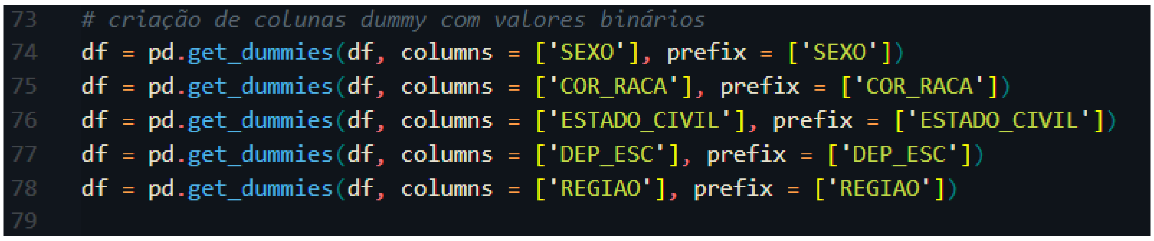


```python
# criação de dicionário
sexo = {'M': 1, 'F': 0}
escolaridade = {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7, 'H': 0}
profissao = {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 0}
regiao = {1: 'N', 2: 'NE', 3:'SE', 4:'S', 5:'CO'}
renda = {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8, 'J': 9,
         'K': 10, 'L': 11, 'M': 12, 'N': 13, 'O': 14, 'P': 15, 'Q': 16}
possui_carro_compu = {'A': 0, 'B': 1, 'C': 1, 'D': 1, 'E': 1}
possui_inter = {'A': 0, 'B': 1}

# aplicação do mapeamento definido no dicionário
df['SEXO'] = df['SEXO'].map(sexo)
df['ESC_PAI'] = df['ESC_PAI'].map(escolaridade)
df['ESC_MAE'] = df['ESC_MAE'].map(escolaridade)
df['PRF_PAI'] = df['PRF_PAI'].map(profissao)
df['PRF_MAE'] = df['PRF_MAE'].map(profissao)
df['REGIAO'] = df['REGIAO'].map(regiao)
df['RENDA_MENSAL'] = df['RENDA_MENSAL'].map(renda)
df['POSSUI_CARRO'] = df['POSSUI_CARRO'].map(possui_carro_compu)
df['POSSUI_COMPU'] = df['POSSUI_COMPU'].map(possui_carro_compu)
df['POSSUI_INTER'] = df['POSSUI_INTER'].map(possui_inter)
```

Source: Bueno and Boente (2024).

This approach allows us to preserve the information contained in the categorical variables without imposing a hierarchical order between the categories. Fig. 5 illustres that creates dammy column for binary values.

Fig. 5 - Excerpt from the program that creates dammy column for binary values



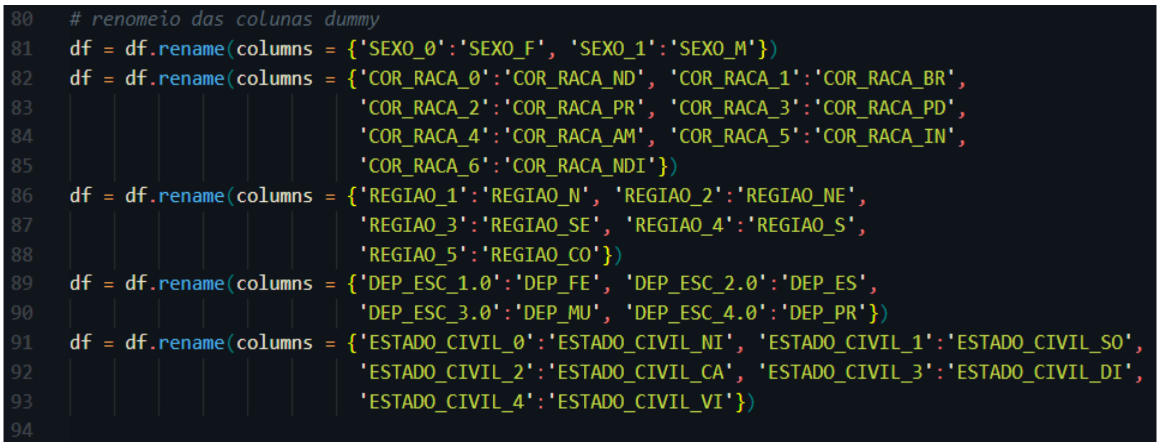
```python
# criação de colunas dummy com valores binários
df = pd.get_dummies(df, columns = ['SEXO'], prefix = ['SEXO'])
df = pd.get_dummies(df, columns = ['COR_RACA'], prefix = ['COR_RACA'])
df = pd.get_dummies(df, columns = ['ESTADO_CIVIL'], prefix = ['ESTADO_CIVIL'])
df = pd.get_dummies(df, columns = ['DEP_ESC'], prefix = ['DEP_ESC'])
df = pd.get_dummies(df, columns = ['REGIAO'], prefix = ['REGIAO'])
```

Source: Bueno and Boente (2024).

Finally, we renamed the created binary columns to more clearly and intuitively reflect the original categories of the categorical variables, as illustrated in Fig. 6.

We split the dataset into ‘X’ for the input variables (features) and ‘y’ for the output variable (the one we want to predict). We also used the Scikit-learn train_test_split() function to split the input variables ‘X’ and the output variable ‘y’ into training (X_train and y_train) and test (X_test and y_test) sets. We set the test_size parameter to 0.25, indicating that 25% of the data will be reserved for testing, while 75% will be used for model training. Fig. 7 illustrates the data for model training.

Fig. 6 - Excerpt from the program that renames the dummy columns

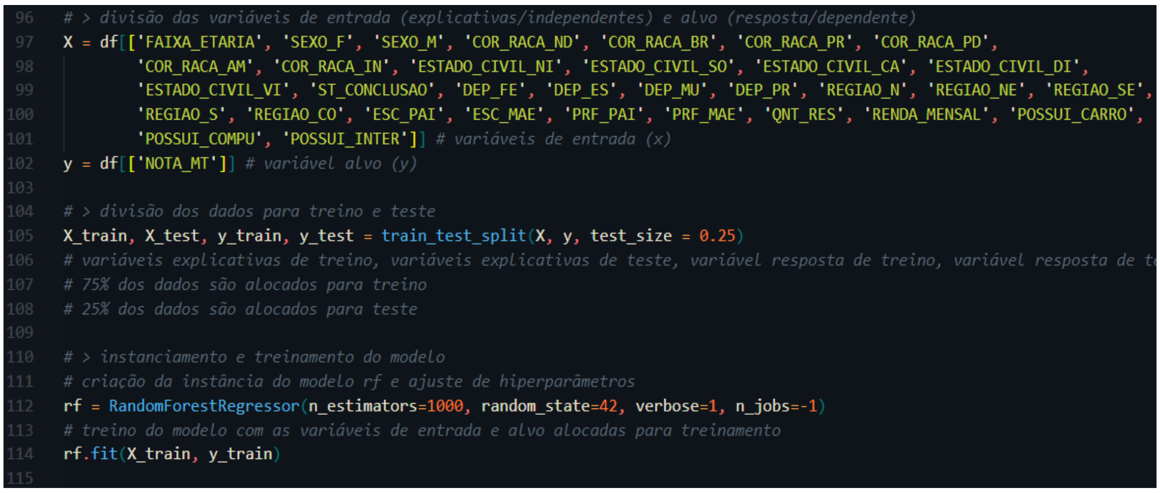


```
# renomeio das colunas dummy
df = df.rename(columns = {'SEXO_0':'SEXO_F', 'SEXO_1':'SEXO_M'})
df = df.rename(columns = {'COR_RACA_0':'COR_RACA_ND', 'COR_RACA_1':'COR_RACA_BR',
                          'COR_RACA_2':'COR_RACA_PR', 'COR_RACA_3':'COR_RACA_PD',
                          'COR_RACA_4':'COR_RACA_AM', 'COR_RACA_5':'COR_RACA_IN',
                          'COR_RACA_6':'COR_RACA_NDI'})
df = df.rename(columns = {'REGIAO_1':'REGIAO_N', 'REGIAO_2':'REGIAO_NE',
                          'REGIAO_3':'REGIAO_SE', 'REGIAO_4':'REGIAO_S',
                          'REGIAO_5':'REGIAO_CO'})
df = df.rename(columns = {'DEP_ESC_1.0':'DEP_FE', 'DEP_ESC_2.0':'DEP_ES',
                          'DEP_ESC_3.0':'DEP_MU', 'DEP_ESC_4.0':'DEP_PR'})
df = df.rename(columns = {'ESTADO_CIVIL_0':'ESTADO_CIVIL_NI', 'ESTADO_CIVIL_1':'ESTADO_CIVIL_SO',
                          'ESTADO_CIVIL_2':'ESTADO_CIVIL_CA', 'ESTADO_CIVIL_3':'ESTADO_CIVIL_DI',
                          'ESTADO_CIVIL_4':'ESTADO_CIVIL_VI'})
```

Source: Bueno and Boente (2024).

Next, we instantiate the Scikit-learn RandomForestRegressor class using the ‘rf’ variable to create the regression model.

Fig. 7 - Excerpt from the program that divides the data for model training



```
# > divisão das variáveis de entrada (explicativas/independentes) e alvo (resposta/dependente)
X = df[['FAIXA_ETARIA', 'SEXO_F', 'SEXO_M', 'COR_RACA_ND', 'COR_RACA_BR', 'COR_RACA_PR', 'COR_RACA_PD',
        'COR_RACA_AM', 'COR_RACA_IN', 'ESTADO_CIVIL_NI', 'ESTADO_CIVIL_SO', 'ESTADO_CIVIL_CA', 'ESTADO_CIVIL_DI',
        'ESTADO_CIVIL_VI', 'ST_CONCLUSAO', 'DEP_FE', 'DEP_ES', 'DEP_MU', 'DEP_PR', 'REGIAO_N', 'REGIAO_NE', 'REGIAO_SE',
        'REGIAO_S', 'REGIAO_CO', 'ESC_PAI', 'ESC_MAE', 'PRF_PAI', 'PRF_MAE', 'QNT_RES', 'RENDA_MENSAL', 'POSSUI_CARRO',
        'POSSUI_COMPU', 'POSSUI_INTER']] # variáveis de entrada (x)
y = df[['NOTA_MT']] # variável alvo (y)

# > divisão dos dados para treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
# variáveis explicativas de treino, variáveis explicativas de teste, variável resposta de treino, variável resposta de te
# 75% dos dados são alocados para treino
# 25% dos dados são alocados para teste

# > instanciamento e treinamento do modelo
# criação da instância do modelo rf e ajuste de hiperparâmetros
rf = RandomForestRegressor(n_estimators=1000, random_state=42, verbose=1, n_jobs=-1)
# treino do modelo com as variáveis de entrada e alvo alocadas para treinamento
rf.fit(X_train, y_train)
```

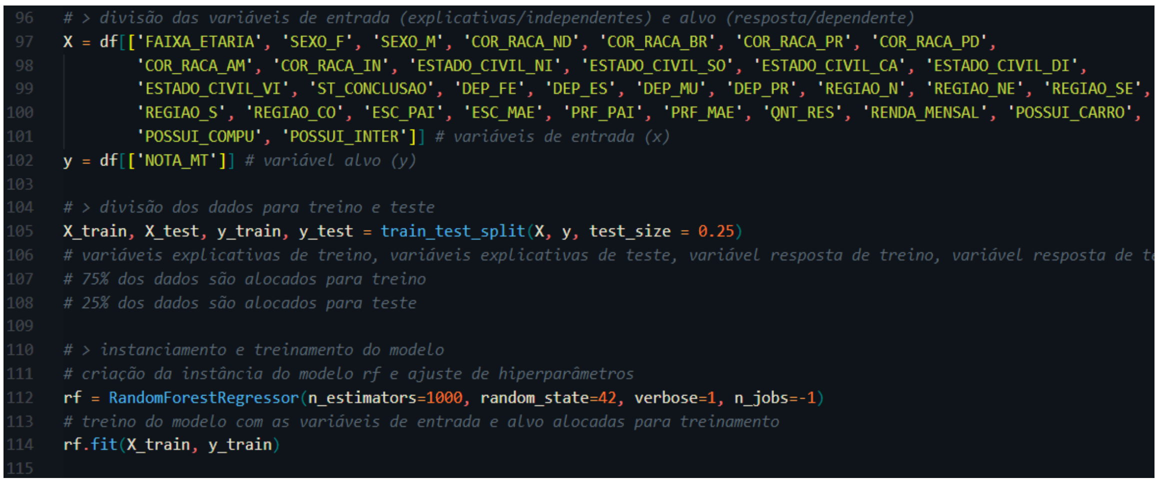
Source: Bueno and Boente (2024).

After training the regression model, we proceed to generate predictions using the test data. This step allows us to evaluate the performance of the model when making predictions on data that was not used during training.

Next, it is essential to evaluate the performance of the model using appropriate metrics. Two common metrics for evaluating regression models are the coefficient of determination (R²) and the mean squared error (MSE). We use the r2_score and mean_squared_error functions in Scikit-learn to calculate these metrics.

The coefficient of determination (R^2) is a measure of the proportion of the variance of the dependent variable that is explained by the independent variables in the model. It ranges from 0 to 1, where 1 indicates a perfect fit of the model to the data. Fig. 8 illustrates the model performance evaluation metric.

Fig. 8 - Excerpt from the program for model performance evaluation metric



```python
# > divisão das variáveis de entrada (explicativas/independentes) e alvo (resposta/dependente)
X = df[['FAIXA_ETARIA', 'SEXO_F', 'SEXO_M', 'COR_RACA_ND', 'COR_RACA_BR', 'COR_RACA_PR', 'COR_RACA_PD',
        'COR_RACA_AM', 'COR_RACA_IN', 'ESTADO_CIVIL_NI', 'ESTADO_CIVIL_SO', 'ESTADO_CIVIL_CA', 'ESTADO_CIVIL_DI',
        'ESTADO_CIVIL_VI', 'ST_CONCLUSAO', 'DEP_FE', 'DEP_ES', 'DEP_MU', 'DEP_PR', 'REGIAO_N', 'REGIAO_NE', 'REGIAO_SE',
        'REGIAO_S', 'REGIAO_CO', 'ESC_PAI', 'ESC_MAE', 'PRF_PAI', 'PRF_MAE', 'QNT_RES', 'RENDA_MENSAL', 'POSSUI_CARRO',
        'POSSUI_COMPU', 'POSSUI_INTER']] # variáveis de entrada (x)
y = df[['NOTA_MT']] # variável alvo (y)

# > divisão dos dados para treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
# variáveis explicativas de treino, variáveis explicativas de teste, variável resposta de treino, variável resposta de te
# 75% dos dados são alocados para treino
# 25% dos dados são alocados para teste

# > instanciamento e treinamento do modelo
# criação da instância do modelo rf e ajuste de hiperparâmetros
rf = RandomForestRegressor(n_estimators=1000, random_state=42, verbose=1, n_jobs=-1)
# treino do modelo com as variáveis de entrada e alvo alocadas para treinamento
rf.fit(X_train, y_train)
```

Source: Bueno and Boente (2024).

The mean squared error (MSE) is the average of the squared errors between the model predictions and the actual values. It provides a measure of the spread of predictions relative to the actual values, with lower values indicating a better fit of the model.

Artificial Neural Nerworks Model

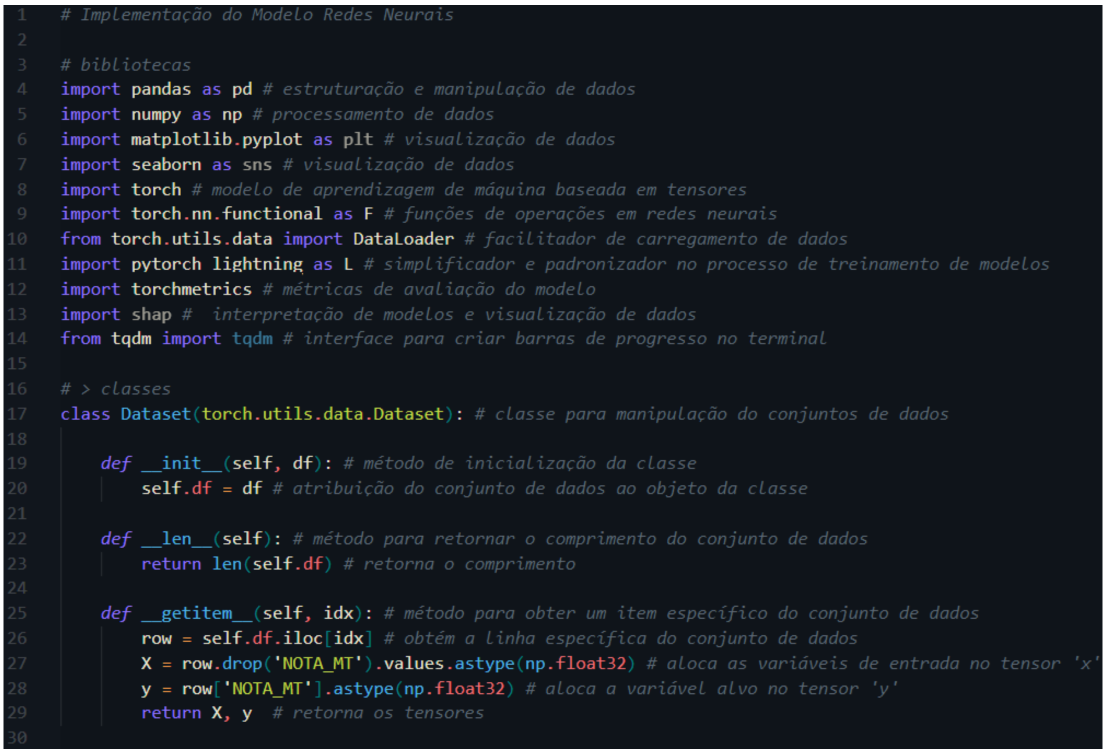
The following libraries were used to implement the Neural Network model, its structuring, manipulation and visualization of data, as well as to apply the machine learning model: Pandas, NumPy, Seaborn, Matplotlib, Torch, Lightning, TorchMetrics, Sharp and Tqdm.

As a means of organizing and encapsulating the code related to the neural network architecture, its parameters, training and inference methods, we created two classes, ‘Dataset’ and ‘Model’.

The Dataset class will be responsible for providing the data for training and validating the neural network model, inheriting functionalities from the torch.utils.data.Dataset class, as illustrated in Fig. 9.

Additionally, Fig. 10 illustrates that the Model class is responsible for the architecture of the neural network and definition of the methods for training, validation and optimization of the model, with functionalities inherited from the L.LightningModule class.

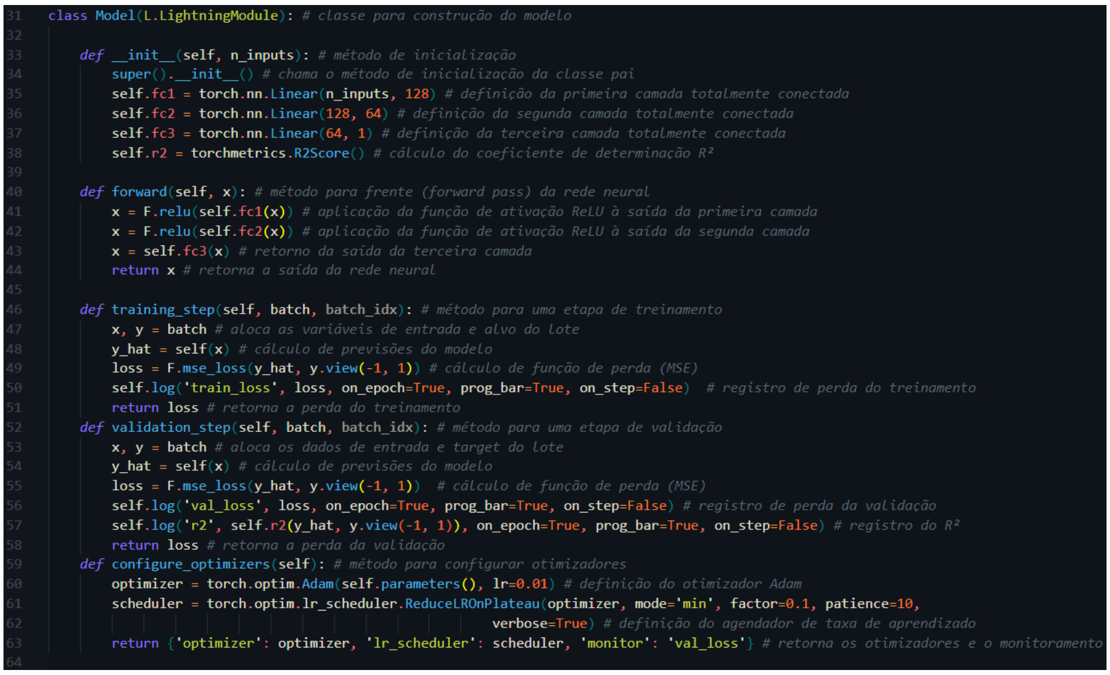
Fig. 9 - Excerpt from the program that defines the Dataset class



```python
# Implementação do Modelo Redes Neurais

# bibliotecas
import pandas as pd # estruturação e manipulação de dados
import numpy as np # processamento de dados
import matplotlib.pyplot as plt # visualização de dados
import seaborn as sns # visualização de dados
import torch # modelo de aprendizagem de máquina baseada em tensores
import torch.nn.functional as F # funções de operações em redes neurais
from torch.utils.data import DataLoader # facilitador de carregamento de dados
import pytorch lightning as L # simplificador e padronizador no processo de treinamento de modelos
import torchmetrics # métricas de avaliação do modelo
import shap #  interpretação de modelos e visualização de dados
from tqdm import tqdm # interface para criar barras de progresso no terminal

# > classes
class Dataset(torch.utils.data.Dataset): # classe para manipulação do conjuntos de dados

    def __init__(self, df): # método de inicialização da classe
        self.df = df # atribuição do conjunto de dados ao objeto da classe

    def __len__(self): # método para retornar o comprimento do conjunto de dados
        return len(self.df) # retorna o comprimento

    def __getitem__(self, idx): # método para obter um item específico do conjunto de dados
        row = self.df.iloc[idx] # obtém a linha específica do conjunto de dados
        X = row.drop('NOTA_MT').values.astype(np.float32) # aloca as variáveis de entrada no tensor 'x'
        y = row['NOTA_MT'].astype(np.float32) # aloca a variável alvo no tensor 'y'
        return X, y  # retorna os tensores

```

Source: Bueno and Boente (2024).

Fig. 10 - Excerpt from the program that defines the Model class and performs data ingestion



```python
class Model(L.LightningModule): # classe para construção do modelo

    def __init__(self, n_inputs): # método de inicialização
        super().__init__() # chama o método de inicialização da classe pai
        self.fc1 = torch.nn.Linear(n_inputs, 128) # definição da primeira camada totalmente conectada
        self.fc2 = torch.nn.Linear(128, 64) # definição da segunda camada totalmente conectada
        self.fc3 = torch.nn.Linear(64, 1) # definição da terceira camada totalmente conectada
        self.r2 = torchmetrics.R2Score() # cálculo do coeficiente de determinação R²

    def forward(self, x): # método para frente (forward pass) da rede neural
        x = F.relu(self.fc1(x)) # aplicação da função de ativação ReLU à saída da primeira camada
        x = F.relu(self.fc2(x)) # aplicação da função de ativação ReLU à saída da segunda camada
        x = self.fc3(x) # retorno da saída da terceira camada
        return x # retorna a saída da rede neural

    def training_step(self, batch, batch_idx): # método para uma etapa de treinamento
        x, y = batch # aloca as variáveis de entrada e alvo do lote
        y_hat = self(x) # cálculo de previsões do modelo
        loss = F.mse_loss(y_hat, y.view(-1, 1)) # cálculo de função de perda (MSE)
        self.log('train_loss', loss, on_epoch=True, prog_bar=True, on_step=False)  # registro de perda do treinamento
        return loss # retorna a perda do treinamento
    def validation_step(self, batch, batch_idx): # método para uma etapa de validação
        x, y = batch # aloca os dados de entrada e target do lote
        y_hat = self(x) # cálculo de previsões do modelo
        loss = F.mse_loss(y_hat, y.view(-1, 1))  # cálculo de função de perda (MSE)
        self.log('val_loss', loss, on_epoch=True, prog_bar=True, on_step=False) # registro de perda da validação
        self.log('r2', self.r2(y_hat, y.view(-1, 1)), on_epoch=True, prog_bar=True, on_step=False) # registro do R²
        return loss # retorna a perda da validação
    def configure_optimizers(self): # método para configurar otimizadores
        optimizer = torch.optim.Adam(self.parameters(), lr=0.01) # definição do otimizador Adam
        scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.1, patience=10,
                                                                verbose=True) # definição do agendador de taxa de aprendizado
        return {'optimizer': optimizer, 'lr_scheduler': scheduler, 'monitor': 'val_loss'} # retorna os otimizadores e o monitoramento

```

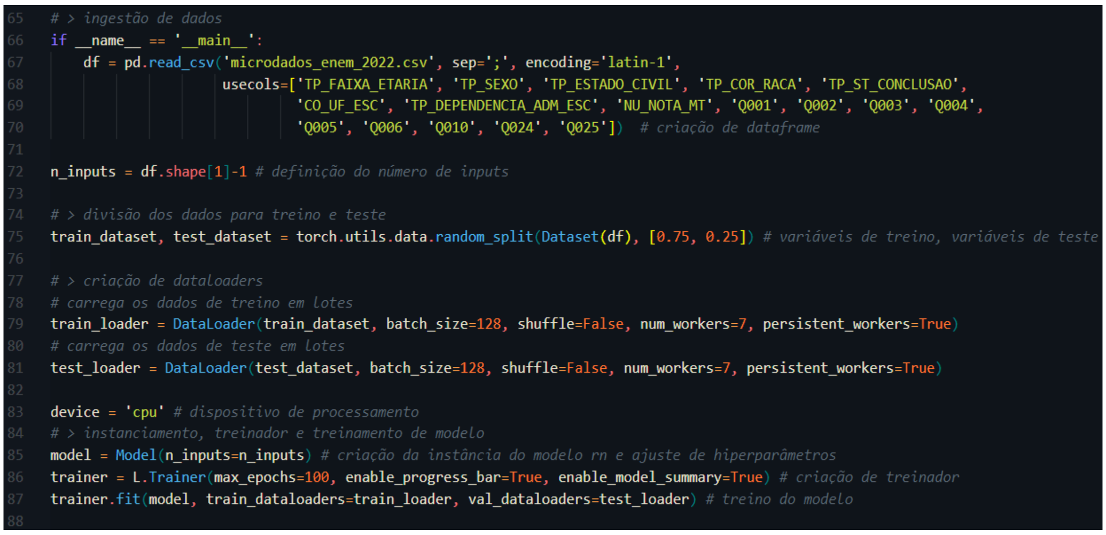
Source: Bueno and Boente (2024).

Next, we declare the condition if __name__ == '__main__':, which checks whether the script is being executed as the main program. If the condition is met, then we perform data ingestion. The ingestion is performed in the same way as when we used the Random Forest model, reproducing the same treatment on the data.

We define the variable n_inputs as the number of columns in the dataframe (df) minus 1, since the last column is considered the target variable. Finally, we use the torch.utils.data.random_split() function to divide the dataset into training and testing sets, where the first argument of the function, the Dataset(df) object, contains all the data, and the second argument indicates the proportion of data that should be attributed to the training set and the testing set (75% for training and 25% for testing).

We create training dataloaders (train_loader) to load the training data (train_dataset) and test dataloaders (test_loader) to load the test data (test_dataset). We define a variable that has the type of device used for model processing and create an instance of the 'Model' model, passing the necessary parameters. Fig. 11 illustres the ingests data and trains the model.

Fig. 11 - Excerpt from the program that ingests data and trains the model

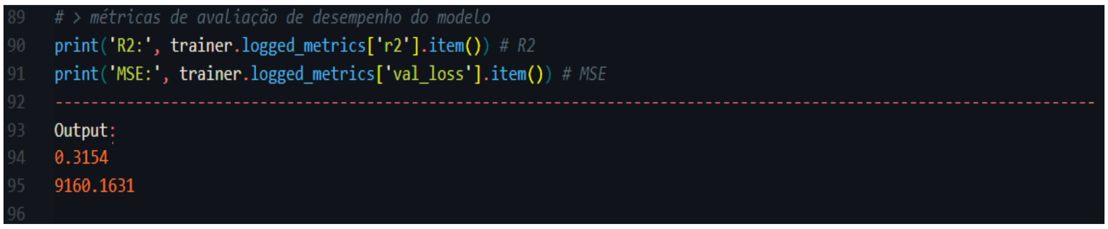


```python
# > ingestão de dados
if __name__ == '__main__':
    df = pd.read_csv('microdados_enem_2022.csv', sep=';', encoding='latin-1',
                     usecols=['TP_FAIXA_ETARIA', 'TP_SEXO', 'TP_ESTADO_CIVIL', 'TP_COR_RACA', 'TP_ST_CONCLUSAO',
                              'CO_UF_ESC', 'TP_DEPENDENCIA_ADM_ESC', 'NU_NOTA_MT', 'Q001', 'Q002', 'Q003', 'Q004',
                              'Q005', 'Q006', 'Q010', 'Q024', 'Q025'])  # criação de dataframe

n_inputs = df.shape[1]-1 # definição do número de inputs

# > divisão dos dados para treino e teste
train_dataset, test_dataset = torch.utils.data.random_split(Dataset(df), [0.75, 0.25]) # variáveis de treino, variáveis de teste

# > criação de dataloaders
# carrega os dados de treino em lotes
train_loader = DataLoader(train_dataset, batch_size=128, shuffle=False, num_workers=7, persistent_workers=True)
# carrega os dados de teste em lotes
test_loader = DataLoader(test_dataset, batch_size=128, shuffle=False, num_workers=7, persistent_workers=True)

device = 'cpu' # dispositivo de processamento
# > instanciamento, treinador e treinamento de modelo
model = Model(n_inputs=n_inputs) # criação da instância do modelo rn e ajuste de hiperparâmetros
trainer = L.Trainer(max_epochs=100, enable_progress_bar=True, enable_model_summary=True) # criação de treinador
trainer.fit(model, train_dataloaders=train_loader, val_dataloaders=test_loader) # treino do modelo
```

Source: Bueno and Boente (2024).

Additionally, Fig. 12 illustrates the model performance evaluation metric.

Fig. 12 - Excerpt from the program for model performance evaluation metric



```
# > métricas de avaliação de desempenho do modelo
print('R2:', trainer.logged_metrics['r2'].item()) # R2
print('MSE:', trainer.logged_metrics['val_loss'].item()) # MSE
----------------------------------------------------------------------------------------------------------------

Output:
0.3154
9160.1631
```

Source: Bueno and Boente (2024).

As shown in the Random Forest model, we will apply the R² and MSE metrics so that we can verify the behavior of the data adjustment in the model and its accuracy in predicting results. We consider that the closer the R² value is to 1, the better the model fits the data and that the lower the MSE value, the better the model is at making accurate predictions.

We consider that the closer the R² value is to 1, the better the model fits the data and that the lower the MSE value, the better the model is at making accurate predictions. Let's look at the comparison between the models presented in Table 1.

Table 1 - Comparing metrics from machine learning models

Metric	Algorithm	Final Result
R2	Random Forest	0.2152369100923196
	Artificial Neural Networks	0.3154
MSE	Random Forest	10602.490903688475
	Artificial Neural Networks	9160.1631

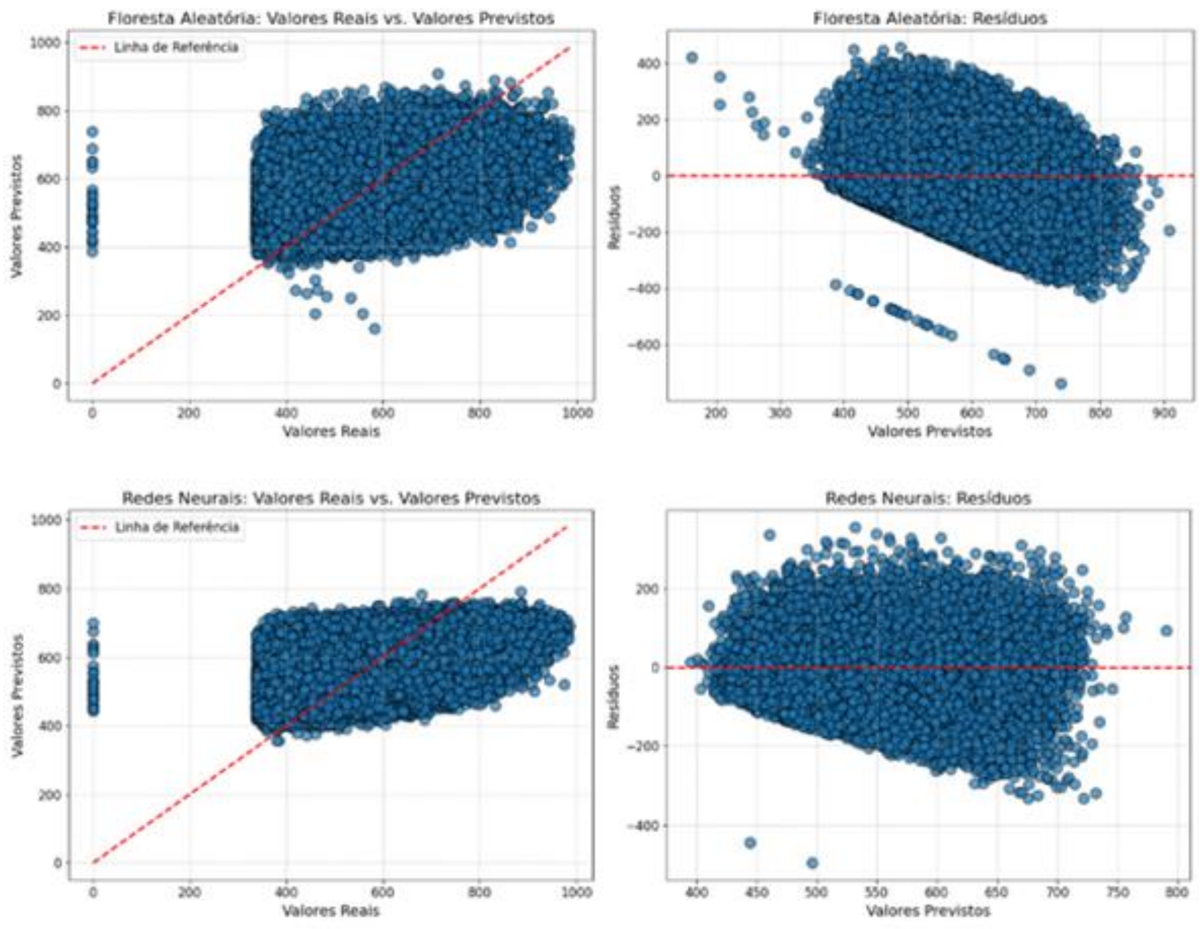
Source: Bueno and Boente (2024).

When comparing the information, we can see that the Random Forest model resulted in an R2 of approximately 0.2152, while the Neural Network model presented a result of 0.3154. This means that only 21.5% of the variability in the candidates' math scores can be explained by the independent variables used in the Random Forest model, indicating that it has a limited explanatory capacity. The Neural Network model, on the other hand, managed to explain 31.54% of the data, demonstrating a better performance in adjusting the trained data.

Regarding the MSE, the Random Forest presented a value of approximately 10602, indicating a significantly high value. This means that the differences between the actual scores and the scores predicted by the model are large, and the model is not accurately predicting the math scores, resulting in estimates that are, on average, quite far from the actual values. In relation to the accuracy result of the previous model, the Neural Networks model was superior, although the MSE value of 9160 suggests the accuracy of the model.

In Fig. 13, the Random Forest model can be analyzed, with a scattering of points around the reference line, indicating that there is a significant amount of prediction errors, that is, the model tends to underestimate or overestimate the real values. In addition, there is a concentration of points in a specific range of real values (between 400 and 800), suggesting that the model may have difficulty predicting correctly outside this range. As for the residual plot, the distribution varies widely, with some distant points, showing inaccuracies and errors in the model.

Fig. 13 - Graphs based on Random Forest and Neural Network models



Source: Bueno and Boente (2024).

When analyzing the performance of the Neural Network model compared to the Random Forest model, a smaller dispersion of points around the reference line in the prediction versus actual values graph is observed. This smaller dispersion suggests that the predictions made by Neural Networks tend to be slightly more accurate, presenting less variation in relation to the observed values. However, it is still possible to identify considerable variability in the results, especially in cases where the actual values are lower. This behavior indicates that the Neural Network model still finds it difficult to accurately predict observations of smaller magnitude, which may impact its generalization capacity in contexts with asymmetric or imbalanced data.

Regarding the residuals graph, a more homogeneous and centralized distribution around the reference line is noted, especially when compared to that observed in the Random Forest model. This characteristic shows a reduction in the model's systematic errors, which can be interpreted as an improvement in the stability of the predictions. The uniformity in the dispersion of the residuals suggests that the model makes errors in a more balanced way throughout the entire range of values, which is desirable in predictive analyses. Based on these aspects, it is concluded that, despite the persistence of significant errors, particularly in lower value ranges, the Neural Networks model outperformed the Random Forest model both in terms of accuracy and consistency of predictions. This demonstrates the greater capacity of the neural model to capture complex patterns in the data, reinforcing its potential for more robust applications in predictive analysis scenarios.

CONCLUSION

The purpose of this study was to apply and compare two machine learning techniques, Random Forest and Artificial Neural Networks, using open microdata from ENEM for the year 2022. The proposal consisted of evaluating the performance of each model in the analysis and interpretation of the correlation between socioeconomic and educational variables and the results obtained in the Mathematics and its Technologies test. In addition, we sought to understand which of the algorithms would be more efficient in generating predictions after the training process, considering criteria such as accuracy and explanatory capacity of the data. At the end of the research, it was observed that, although both models presented limitations in explaining the variation in participants' grades, as well as less than ideal predictive accuracy, Neural Networks demonstrated superior performance in relation to Random Forest. Specifically, the Neural Network model was more efficient in the way it used independent variables to establish interpretable relationships and generate more consistent predictions. This greater capacity for generalization and non-linear analysis contributed to more satisfactory results in predicting candidates' math grades.

Therefore, it is concluded that, within the context analyzed, Neural Networks proved to be more appropriate both for understanding the dynamics between the factors analyzed and for making future inferences. Despite the challenges faced by both models, especially given the complexity of educational data, the use of Neural Networks proved to be a promising alternative for predictive analysis in the field of educational assessment, contributing to a deeper understanding of student performance in ENEM 2022.

REFERENCES

1. Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140.

2. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.

3. Bueno, M. L., & Boente, A. N. P. (2024). *Determinantes de desempenho no ENEM: Uma abordagem em machine learning* [Unpublished bachelor's thesis]. Universidade Veiga de Almeida, Rio de Janeiro, RJ, Brazil.

4. Bueno, M. L., & Boente, A. N. P. (2024, November 11–14). Inteligência artificial e machine learning: Um estudo comparativo dos modelos de aprendizagem de máquina floresta aleatória e redes neurais. In *XXXI SIMPEP - Simpósio de Engenharia de Produção*, Bauru, SP, Brazil.

5. Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning*. MIT Press.

6. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira. (2024). *Microdados do ENEM*. https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/microdados/enem

7. Liaw, A., & Wiener, M. (2002). Classification and regression by random forest. *R News*, 2(3), 18–22.

8. Lima, F. A. M., et al. (2024). Modelo de inteligência artificial aplicado à análise de dados de pessoas com deficiência: Utilização de LangChain. *Revista de Gestão e Secretariado*, 15(10), Article e4281. https://doi.org/10.7769/gesec.v15i10.4281

9. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4), 115–133.

10. Napoleão, B. M. (2024). Árvores decisórias. https://ferramentasdaqualidade.org/arvores-decisorias/

11. Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408.

12. Taulli, T. (2020). *Introdução à inteligência artificial: Uma abordagem não técnica*. Novatec.