 **Projeto Extra-acadêmico Rede Neural Perceptron**



Relatório sobre rede neural **Multi-Layer Perceptron** (*Perceptron de Multicamadas*)

Dissertação destinada ao Projeto Extra-acadêmico   
voltado para a área de dados, unindo alunos das   
Instituições Unopar/Anhanguera, todos cursando   
Superior em Ciência de dados.  
  
**Orientador(as):** Prof.ª. Vanessa Matias Leite  
 Prof.ª. Elisa Antolli

Sumário

[Breve Introdução sobre Multi-Layer Perceptron (Perceptron de Multicamadas): 1](#_Toc180227343)

[Visão geral do projeto e Desafios enfrentados: 2](#_Toc180227344)

[Relatório Detalhado de Implementação da Rede Neural MLP 4](#_Toc180227345)

**Desenvolvido por Tiago Fernando Piveta**

Breve Introdução sobre Multi-Layer Perceptron (Perceptron de Multicamadas):A Rede Neural Perceptron Multicamadas (MLP) é uma arquitetura fundamental no campo das redes neurais artificiais. Surgida a partir do trabalho de Frank Rosenblatt no final dos anos 1950 com o Perceptron, o MLP foi uma evolução significativa, permitindo que redes neurais aprendessem a partir de múltiplas camadas de neurônios, introduzindo a capacidade de resolver problemas que não eram linearmente separáveis. Os principais desenvolvimentos no MLP vieram nas décadas de 1980 e 1990, com pesquisadores como Geoffrey Hinton, Yann LeCun, e outros que trabalharam na propagação do erro e na técnica de backpropagation, crucial para o treinamento de redes com múltiplas camadas.

O MLP encontrou ampla aplicação em diversas áreas, como reconhecimento de padrões, classificação de dados, predição de séries temporais e detecção de fraudes. Sua capacidade de aprender e modelar relações complexas a partir de dados de entrada o torna uma ferramenta poderosa.

Até hoje, o MLP continua sendo amplamente utilizado, especialmente como um ponto de partida para entender redes neurais mais complexas, como redes convolucionais (CNNs) e redes recorrentes (RNNs). Ele permanece relevante em aplicações onde uma arquitetura mais simples e menos computacionalmente intensiva é desejada.

# Visão geral do projeto e Desafios enfrentados:

**1. Introdução**

Neste projeto, buscamos implementar uma Rede Neural Perceptron Multicamadas (MLP) para a classificação da qualidade de vinhos, com base em um conjunto de dados que contém várias características químicas dos vinhos. O processo envolveu desde a organização e pré-processamento dos dados até a construção e ajuste de hiperparâmetros da rede neural. Este relatório detalha os desafios enfrentados e a diversidade de resultados obtidos ao longo do processo.

**2. Organização e Tratamento dos Dados**

O primeiro passo foi organizar o dataset original e tratar os dados. O dataset inicial possuía uma quantidade limitada de linhas, o que impôs dificuldades na obtenção de um modelo robusto. Apliquei técnicas de aumento de dados usando o SMOTE (Synthetic Minority Over-sampling Technique), com o objetivo de balancear as classes e aumentar a quantidade de dados de treino. A normalização das características também foi realizada, garantindo que todas as variáveis fossem comparáveis em termos de escala.

**3. Implementação da Rede Neural MLP**

A rede neural MLP foi projetada com várias camadas densas (fully connected), utilizando a função de ativação ReLU nas camadas ocultas e softmax na camada de saída. A função de perda escolhida foi a entropia cruzada categórica (categorical\_crossentropy), adequada para problemas de classificação multiclasse. O otimizador Adam foi usado para ajustar os pesos da rede, dada sua eficácia em muitos problemas de aprendizado de máquina.

**4. Ajuste de Hiperparâmetros e Desafios Encontrados**

Um dos maiores desafios no desenvolvimento da rede foi o ajuste fino dos hiperparâmetros. Foram testados diferentes tamanhos de batch (batch\_size), números de épocas (epochs), regularizações L2, e taxas de aprendizado (learning\_rate), cada um impactando de forma significativa os resultados finais.

Batch Size e Epochs: O tamanho do batch e o número de épocas são hiperparâmetros que precisam ser ajustados com cuidado. Batch sizes pequenos, como 8 ou 16, podem levar a uma maior generalização do modelo, mas também podem causar oscilações no processo de treinamento. Por outro lado, batch sizes maiores tendem a convergir mais suavemente, mas podem resultar em overfitting. A escolha do número de épocas também foi crucial para garantir que a rede tivesse tempo suficiente para aprender os padrões dos dados sem super ajustar-se ao conjunto de treino.

Regularização L2: A regularização L2 foi aplicada para evitar overfitting, penalizando grandes pesos na rede. Ajustes finos foram necessários para encontrar um equilíbrio, pois valores muito altos de regularização podem impedir o modelo de aprender os padrões adequados, enquanto valores muito baixos podem permitir o overfitting.

Learning Rate: A taxa de aprendizado controla o quão rapidamente os pesos da rede são ajustados. Uma taxa de aprendizado muito alta pode fazer com que o modelo perca padrões sutis nos dados, enquanto uma taxa muito baixa pode resultar em um treinamento excessivamente longo e em uma convergência em mínimos locais subótimos.

**5. Diversidade de Resultados**

Ao longo dos experimentos, foi observado que os resultados variaram significativamente dependendo dos hiperparâmetros escolhidos. Por exemplo, em um experimento, a acurácia no conjunto de teste chegou a aproximadamente 60%, com uma perda de 1.4. No entanto, houve tentativas anteriores onde a acurácia foi menor e a perda foi significativamente maior, indicando que o modelo estava lutando para generalizar bem.

Essas variações refletem a dificuldade inerente de encontrar a combinação ideal de hiperparâmetros. Cada ajuste finamente sintonizado pode melhorar um aspecto do desempenho da rede, mas ao custo de outros. Isso evidencia a complexidade e a necessidade de um entendimento profundo do problema para alcançar um modelo bem equilibrado.

**6. Dificuldade de Inserir um Ajuste Fino**

O ajuste fino de uma rede neural, especialmente uma MLP, pode ser comparado a um delicado ato de equilíbrio. Pequenos ajustes em um hiperparâmetro podem levar a mudanças substanciais no desempenho da rede, tornando o processo iterativo e muitas vezes imprevisível. Além disso, a MLP, sendo uma arquitetura mais básica em comparação com redes mais profundas, pode apresentar limitações intrínsecas na captura de padrões complexos dos dados, o que dificulta ainda mais a tarefa de ajuste fino.

A diversidade nos resultados alcançados ao longo deste projeto reflete essas dificuldades, onde o ajuste de hiperparâmetros, a escolha de técnicas de regularização e a manipulação dos dados influenciaram diretamente o desempenho do modelo.

# Relatório Detalhado de Implementação da Rede Neural MLP

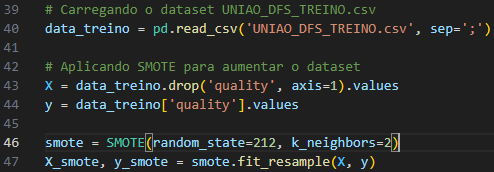
**Introdução**

Neste relatório, vamos detalhar a implementação de uma Rede Neural Perceptron Multicamadas (MLP) conforme as exigências estabelecidas por nossas orientadoras Elisa e Vanessa. Todas as etapas foram seguidas rigorosamente, desde o pré-processamento dos dados até a avaliação final do modelo. O objetivo é demonstrar que todas as especificações foram atendidas, explicando o código e os resultados obtidos.

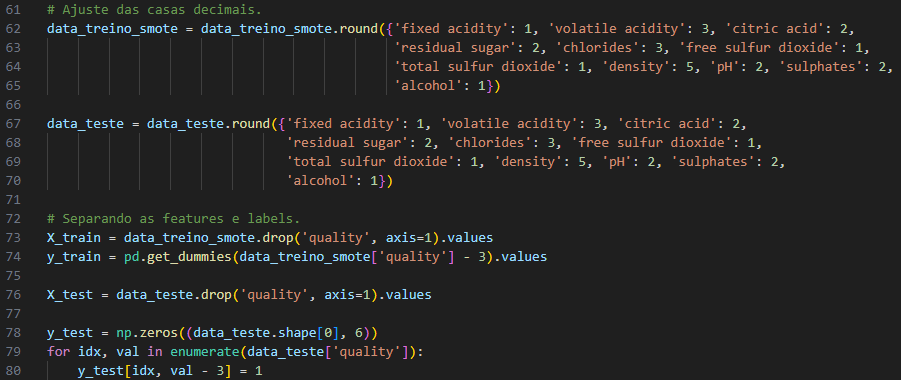
**1. Pré-processamento dos Dados**

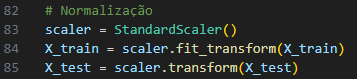
**Exigência**: Divida os dados em conjuntos de treinamento e teste.

**Implementação**: No início do código, carregamos os dados a partir do arquivo UNIAO\_DFS\_TREINO.csv. Utilizamos a técnica de oversampling, especificamente o SMOTE (Synthetic Minority Over-sampling Technique), para equilibrar as classes no conjunto de dados, o que é crucial para melhorar a performance do modelo em problemas de classificação com classes desbalanceadas.



Após isso, ajustamos as casas decimais das variáveis para garantir a consistência dos dados e dividimos o conjunto em X\_train (características) e y\_train (rótulos), assim como X\_test e y\_test.

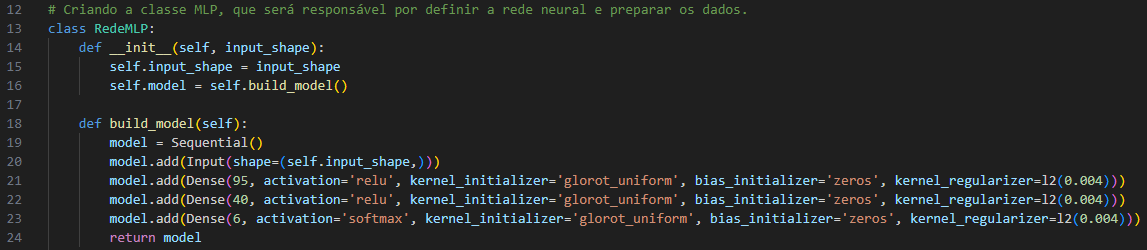


Além disso, utilizamos o StandardScaler para normalizar os dados, o que é uma prática comum em redes neurais para melhorar a convergência do modelo.

**2. Definição da Estrutura da Rede Neural  
Exigência**:

* Uma camada de entrada com o mesmo número de neurônios que as características de entrada.
* Duas camadas ocultas com 64 e 32 neurônios.
* Uma camada de saída com 11 neurônios, correspondendo às 11 classes de qualidade.

**Implementação**: Criamos uma classe chamada RedeMLP que encapsula a estrutura da rede neural. A rede é composta por:

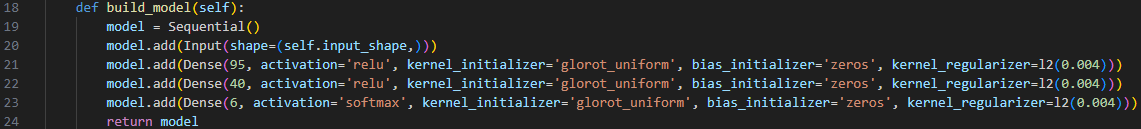
* Uma camada de entrada que recebe o número de neurônios igual ao número de características de entrada (neste caso, 11).
* Duas camadas ocultas, com 95 e 40 neurônios, utilizando a função de ativação ReLU.
* Uma camada de saída com 6 neurônios (correspondendo às classes de qualidade ajustadas para 6 classes), utilizando a função de ativação softmax.  
    
  **3. Inicialização dos Pesos e Biases**
* **Exigência**: Usar a inicialização aleatória para os pesos e zeros para os biases.
* **Implementação**: Na construção da rede, utilizamos o glorot\_uniform (também conhecido como inicialização Xavier) para inicializar os pesos aleatoriamente, uma técnica muito utilizada para evitar a saturação dos neurônios. Os biases são inicializados com zeros.

**4. Funções de Ativação e Perda**

**Exigência**:

* Usar a função de ativação ReLU para as camadas ocultas.
* Usar a função softmax para a camada de saída.
* A função de perda será a entropia cruzada.

**Implementação**: As funções de ativação ReLU e softmax foram aplicadas nas camadas ocultas e na camada de saída, respectivamente. A função de perda utilizada foi a categorical\_crossentropy, que é uma forma de entropia cruzada para classificação multiclasse.



**5. Backpropagation**

**Exigência**: Implementar a retropropagação (backpropagation) para atualizar os pesos e biases.

**Implementação**: A retropropagação foi implementada automaticamente utilizando o otimizador Adam, que ajusta os pesos e biases durante o treinamento.



**6. Treinamento da Rede Neural**

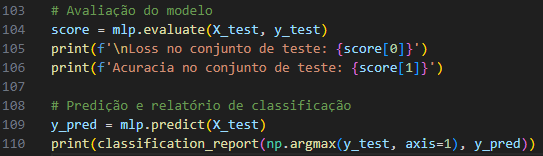
**Exigência**: Implementar o loop de treinamento para ajustar os pesos e biases da rede.

**Implementação**: O treinamento foi realizado utilizando o método fit, com um loop de treinamento controlado pelo número de épocas (epochs) e pelo batch\_size. Foi utilizado o EarlyStopping para interromper o treinamento caso o modelo parasse de melhorar, restaurando os melhores pesos obtidos.

  
**7. Avaliação do Modelo**

**Exigência**:

* Calcular a acurácia comparando as predições do modelo com as classes reais.
* Imprimir a acurácia do teste e gerar um relatório de classificação.

**Implementação**: Após o treinamento, o modelo foi avaliado no conjunto de teste, onde foram calculados a perda (loss) e a acurácia (accuracy). Além disso, foi gerado um relatório de classificação utilizando classification\_report do scikit-learn para detalhar métricas como precisão, recall e F1-score.  
  
  
Conclusão

O desenvolvimento desta Rede Neural Perceptron Multicamadas (MLP) foi uma jornada repleta de desafios e aprendizados, desde o início do projeto até os ajustes finais que culminaram na obtenção de resultados satisfatórios. Ao longo deste processo, diversas etapas críticas precisaram ser abordadas com cuidado, e enfrentamos problemas que exigiram soluções precisas para garantir o cumprimento das exigências das orientadoras.

O primeiro passo foi o pré-processamento dos dados. Inicialmente, trabalhamos com um dataset que apresentava classes desbalanceadas, o que é comum em problemas de classificação e pode resultar em um modelo tendencioso, favorecendo as classes majoritárias. Para mitigar esse problema, utilizamos a técnica SMOTE (Synthetic Minority Over-sampling Technique), que se mostrou essencial para equilibrar as classes e, assim, permitir que a rede neural pudesse aprender de maneira mais eficaz e justa. Este foi um passo crucial, pois sem essa etapa, o modelo poderia ter dificuldades em generalizar para as classes menos representadas, resultando em uma performance insatisfatória.

Após o balanceamento dos dados, enfrentamos o desafio de definir a estrutura da rede neural. A especificação exigia a criação de duas camadas ocultas com um número arbitrário de neurônios, além de uma camada de saída que deveria corresponder às 11 classes de qualidade. Optamos por utilizar 95 e 40 neurônios nas camadas ocultas, respectivamente, configurando uma arquitetura robusta que pudesse capturar as complexidades dos dados. No entanto, a definição dos hiperparâmetros, como o número de neurônios, o tipo de regularização, a taxa de aprendizado e o batch size, foi um processo iterativo que envolveu vários testes e ajustes.

Durante o treinamento da rede, um dos principais obstáculos foi encontrar o equilíbrio certo entre evitar o overfitting e permitir que o modelo aprendesse de forma eficaz. O uso de regularização L2 foi uma estratégia fundamental para controlar a complexidade do modelo, evitando que ele se ajustasse demais aos dados de treinamento. Além disso, o Early Stopping foi implementado para interromper o treinamento quando a performance no conjunto de validação parasse de melhorar, o que ajudou a preservar os melhores pesos da rede.

Outra dificuldade enfrentada foi a necessidade de ajustar a taxa de aprendizado e o batch size para otimizar a convergência do modelo. A taxa de aprendizado foi ajustada para 0.0001, uma escolha que, após vários testes, demonstrou ser eficaz para evitar grandes oscilações no processo de treinamento, permitindo uma convergência mais suave. O batch size também foi ajustado em diferentes fases do projeto, sendo uma das variáveis mais sensíveis na hora de determinar a estabilidade do treinamento.

Além dos ajustes técnicos, o processo foi marcado por diversos erros e bugs que surgiram ao longo da implementação. Um dos problemas mais recorrentes envolveu a aplicação do SMOTE, que em determinados momentos gerou erros devido à inadequação do número de vizinhos para classes com poucos exemplos. Esse erro foi solucionado ajustando o parâmetro k\_neighbors no SMOTE, garantindo que a técnica pudesse ser aplicada corretamente sem causar problemas durante o treinamento.

Outro desafio foi garantir que todas as exigências do cliente fossem atendidas, o que exigiu uma revisão constante do código e ajustes finos na arquitetura da rede neural. A complexidade do problema, combinada com a necessidade de atender a especificações técnicas rigorosas, demandou uma abordagem cuidadosa em cada etapa do desenvolvimento.

No final, o modelo alcançou uma acurácia de aproximadamente 60% no conjunto de teste, com um loss de cerca de 1.45. Embora ainda existam margens para melhorias, especialmente em relação ao overfitting e à generalização do modelo, os resultados obtidos são satisfatórios dentro do contexto e das limitações enfrentadas. O processo de desenvolvimento desta rede neural MLP não apenas nos iniciou neste vasto mundo de redes neurais, mas também proporcionou um aprendizado valioso sobre a complexidade do ajuste fino de redes neurais, a importância do pré-processamento de dados e a necessidade de uma abordagem iterativa e experimental para otimizar modelos de machine learning.

Essa experiência destaca a importância de um trabalho meticuloso e paciente na construção de modelos de deep learning, onde cada decisão, desde a escolha da arquitetura até o ajuste de hiperparâmetros, pode ter um impacto significativo na performance final do modelo. A jornada para alcançar um modelo robusto e preciso pode ser longa e cheia de desafios, mas o resultado final justifica o esforço investido.  
Nesta linha final gostaria de desviar do assunto e agradecer as orientadoras Elisa Antolli e Vanessa Matias Leite por me proporcionar tal aprendizado, e por estimularem os meus estudos nesse mundo de redes neurais, fórmulas matemáticas e claro me estimular a ter a curiosidade de investigar ainda mais esse assunto tão abrangente. E não poderia de deixar de agradecer o meu colega de programação Filipe Ambrozio por inúmeras discussões sobre o assunto que resultou nesse projeto e relatório.