Trabalho Prático 1: Treinamento e Avaliação de Modelos de Aprendizado Supervisionado

Grupo 3

Eduardo Aurélio H. Duarte (311408), Lucas P. Pons (312430),
Paola A. Andrade (306031), Vitoria Lentz(301893)

{paandrade, eahduarte, lppons, vlentz}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul - UFRGS
Instituto de Informática - INF

1. Objetivo

O propósito central de estudo do trabalho foi entender como bancos selecionam potenciais consumidores a partir de informações pessoais como idade, salário, ocupação, estado civil, etc., maximizando a chance de vender um produto. A motivação de prever um potencial consumidor está em reduzir esforços com pessoas que não estão dispostas a consumir o produto vendido. O dataset utilizado foi bank-marketing encontrado no Kaggle.

Para o desenvolvimento dos modelos utilizamos a linguagem de programação Python, com o auxílio de bibliotecas como: *Pandas* (para manuseio do *dataset* durante o pré-processamento dos dados), *matplotlib* e *seaborn* (para visualização das métricas e resultados) e *scikit-learn* (para treinamento e teste dos modelos utilizados).

2. Informações do Dataset

O dataset possui 45221 linhas e 22 colunas. Cada instância é um consumidor com 22 atributos, alguns atributos mais relevantes podem ser vistos na Tabela 1 abaixo. Os atributos *response* e *y*, correspondem à classe de cada instância, representando se um consumidor respondeu positivamente ou não ao produto. Analisando o dataset pode se observar que a classe *y=0* possui 39922 instâncias aos 5299 da classe oposta, como visto na Figura 1.

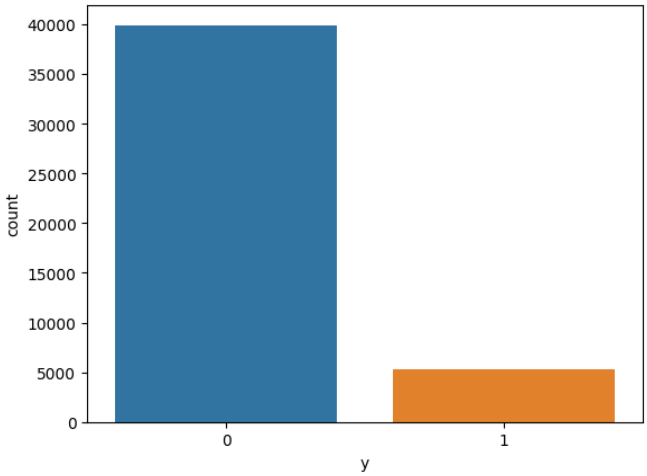

Figura 1- Distribuição de instâncias por classe

atributo	descrição	data type	exemplo
age	idade do consumidor	int	21, 55, 60
eligible	se o consumidor está elegível para conversa ou não	object	Y, N
job	o que o consumidor faz?	object	technician, management
salary	salário do consumidor	int	60000, 120000
marital	estado civil do consumidor	object	single, married, divorced
education	nível de educação	object	primary, tertiary, unknown
balance	saldo do consumidor	int	2432, 29
housing	se o consumidor possui habitação	object	yes, no
response	resposta positiva ou negativa do consumidor	int	1, 0

Tabela 1 - Descrição dos atributos do dataset

3. Metodologia

A metodologia aplicada ao trabalho consiste em:

1. Análise, balanceamento e pré-processamento dos dados
2. Normalização de atributos numéricos
3. Aplicação da técnica de K-folds para validação cruzada
4. Escolha de algoritmos (modelos) e hiperparâmetros
5. Análise dos resultados

4. Data Processing

Encoding & Scaling

A primeira parte do pré-processamento de dados consistiu em converter atributos categóricos para inteiros. As funções usadas para *encoding* variam para cada atributo: TargetEncoder para atributos com muitas categorias como ocupação, OneHotEncoder para atributos binários ou com poucas categorias como educação e estado civil. Além disso, atributos numéricos como idade e salário foram normalizados utilizando a função StandardScaler da biblioteca sklearn. Algumas colunas consideradas irrelevantes, como *age group*, foram retiradas completamente dos dados

Balanceamento dos Dados

Como mencionado anteriormente, o dataset original é muito desbalanceado, com uma razão de aproximadamente 1:8. Foi comparado a performance de cada modelo com os dados balanceados pela função StandardScaler e desbalanceados, a comparação pode ser vista na Figura 3 na seção 5. Concluímos que balancear o dataset produz melhores resultados.

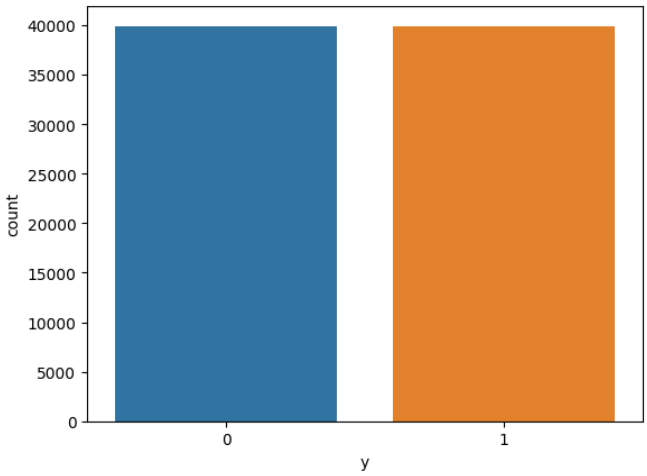

Figura 2 - Número de instâncias por classe após o balanceamento

5. K-fold

Após o processamento dos dados, acontece a divisão dos dados entre teste e treino. Para isso, utilizamos o mecanismo de validação cruzada K-fold (com o auxílio da biblioteca *sckit-learn*), que provê confiabilidade de resultados, sem vieses, uma vez que sua metodologia de particionamento dos dados garante que toda instância do *dataset* servirá tanto para treino quanto para teste.

Neste trabalho, utilizamos a função KFold da biblioteca *sckit-learn* para realizar o particionamento dos dados, arbitrando que a quantidade de folds seria k=19, variando de 2 a 20. A linha abaixo, retirada do arquivo `validation.py`, exemplifica a utilização da função mencionada:

```
# K-folds de dados de treino e teste
folds = KFold(n_splits=k, shuffle=True, random_state=7).split(X, y)
```

6. Análise dos Modelos

Para desenvolvimento do trabalho, foram escolhidos os algoritmos k-Nearest Neighbors, Árvores de Decisão e Redes Neurais, pela suposição de que a comparação entre eles seria simplificada, assim como a otimização dos hiperparâmetros e a interpretabilidade de ambos.

O estudo destes modelos começou com a análise dos hiperparâmetros e o balanceamento dos dados, para cada modelo foi avaliado os seguintes parâmetros:

- kNN: o valor de k
- Árvore de Decisão: profundidade máxima da árvore
- Rede Neural: camadas ocultas e número de iterações/épocas

Os modelos foram treinados com dados balanceados e desbalanceados, os resultados são vistos nos gráficos abaixo.

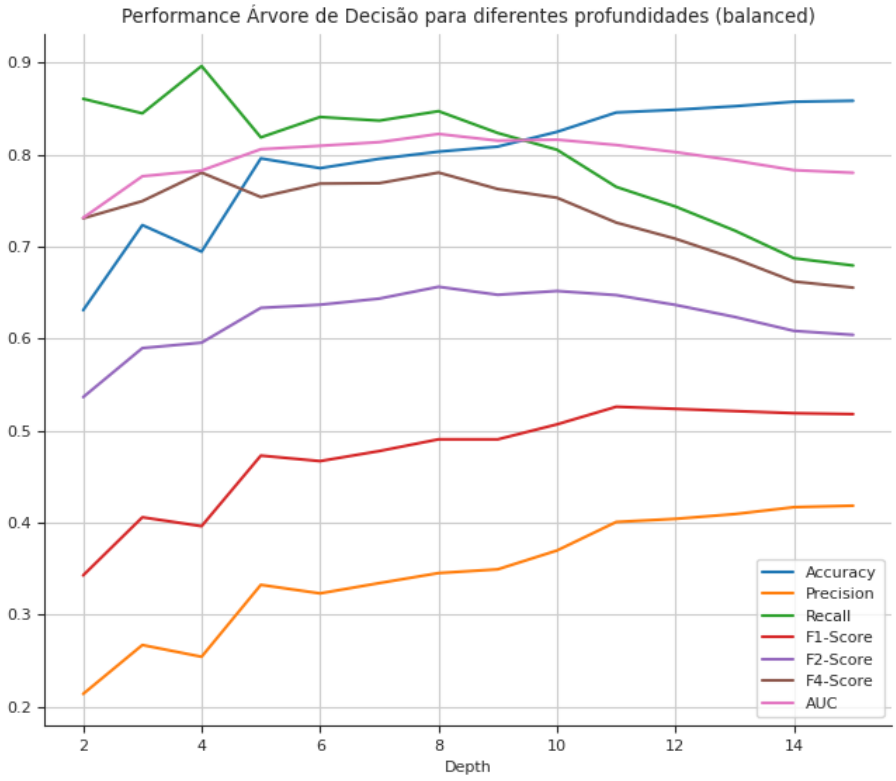

Figura 3 - Gráfico de performance da Árvore de decisão para diferentes profundidades considerando o *dataset* balanceado

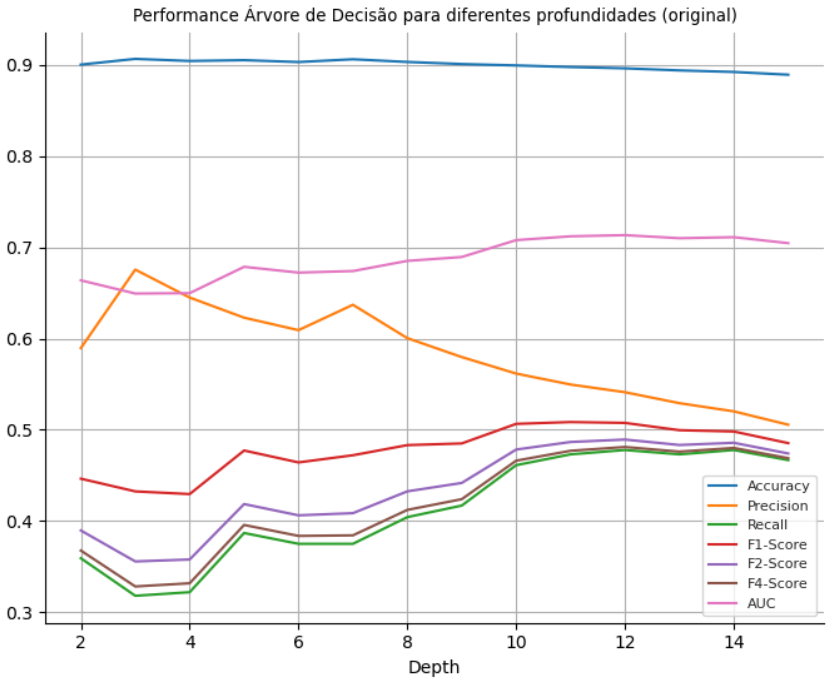

Figura 4 - Gráfico de performance da Árvore de decisão para diferentes profundidades considerando o *dataset* original

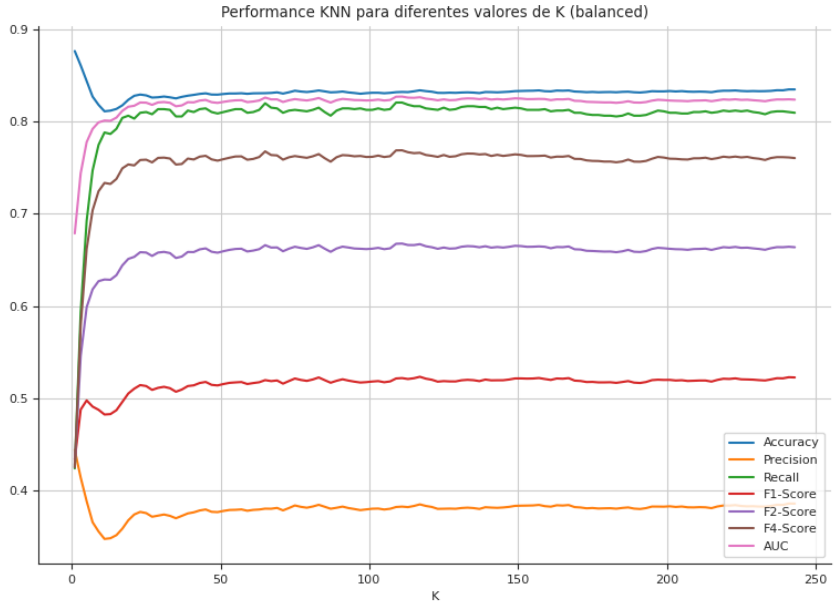

Figura 5 - Gráfico de performance do k-Nearest Neighbors para diferentes profundidades considerando o *dataset* balanceado

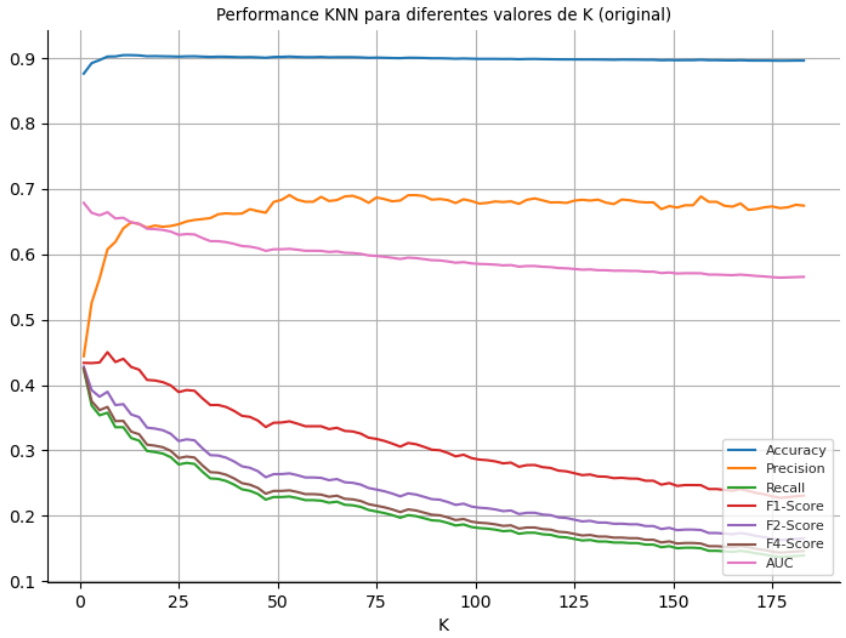

Figura 6 - Gráfico de performance do k-Nearest Neighbors para diferentes profundidades considerando o *dataset* original

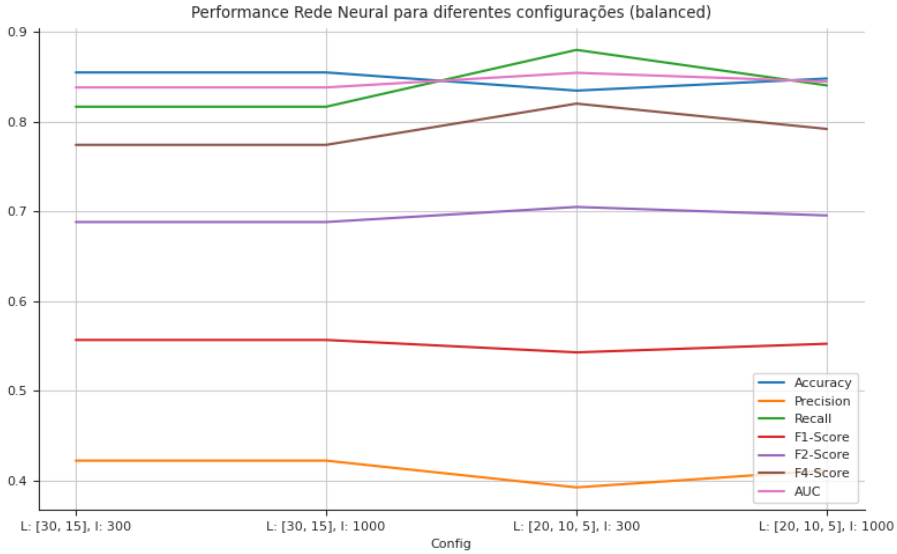

Figura 7 - Gráfico de performance da Rede Neural para diferentes profundidades considerando o *dataset* balanceado

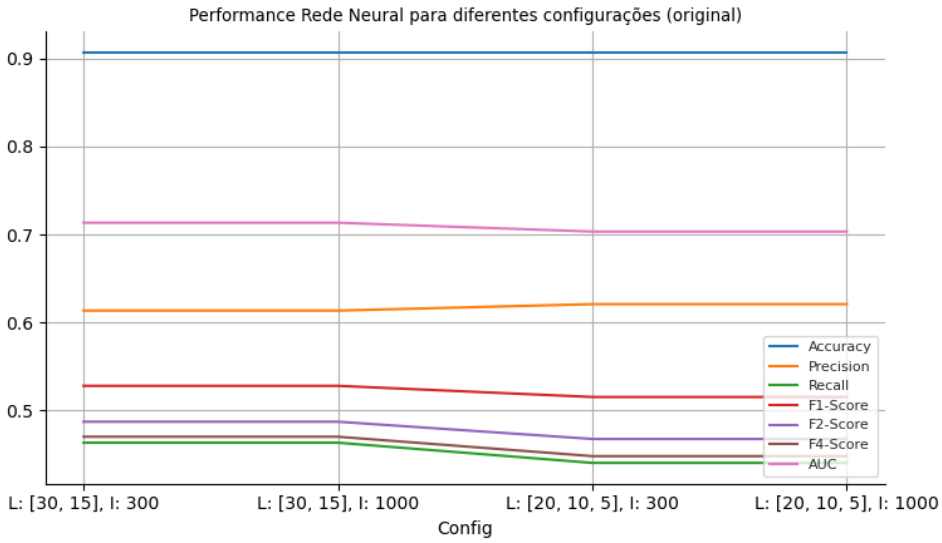

Figura 8 - Gráfico de performance da Rede Neural para diferentes profundidades considerando o *dataset* original

Segundo o contexto do dataset escolhido, um banco prevendo um potencial consumidor não deseja que alguém que compraria o produto vendido seja erradamente classificado, pois isso diminuiria os lucros. Conforme essa lógica, decidimos que a métrica a ser priorizada é o *recall,* minimizando os falsos negativos. Com isso em mente, concluímos que os melhores parâmetros para os modelos foi o seguinte:

- kNN: k igual a 50
- Árvore de Decisão: profundidade máxima igual a 4
- Rede Neural: 3 camadas ocultas com 20, 10 e 5 neurônios respectivamente e 500 iterações

Em seguida foi analisado o melhor k para o k-Fold. O estudo não foi muito significativo, pois alterar o valor de k não resultou em uma mudança de performance considerável. Os gráficos dos testes são vistos abaixo.

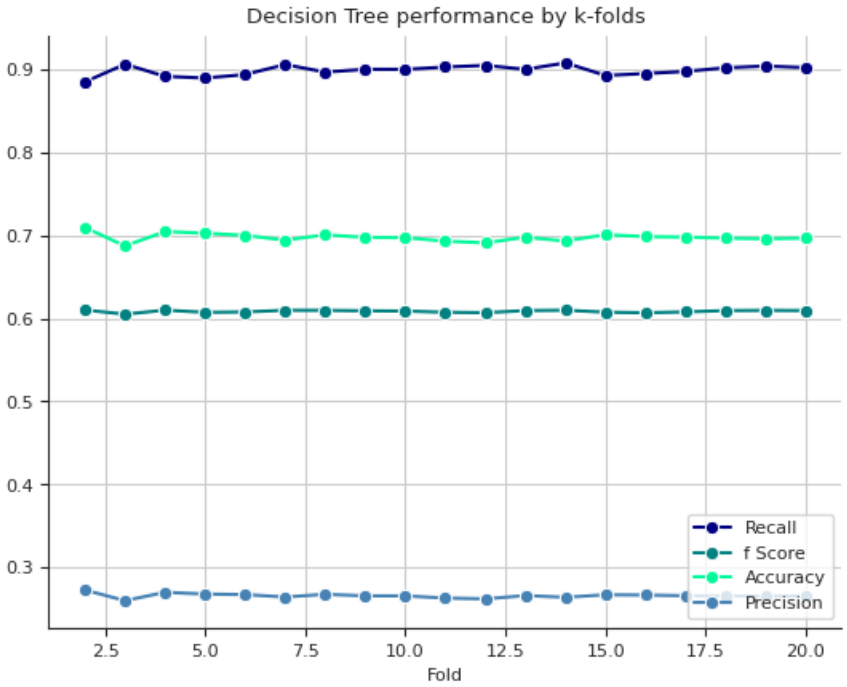

Figura 9 - Gráfico de performance da Árvore de Decisão com validação cruzada utilizando k-Fold

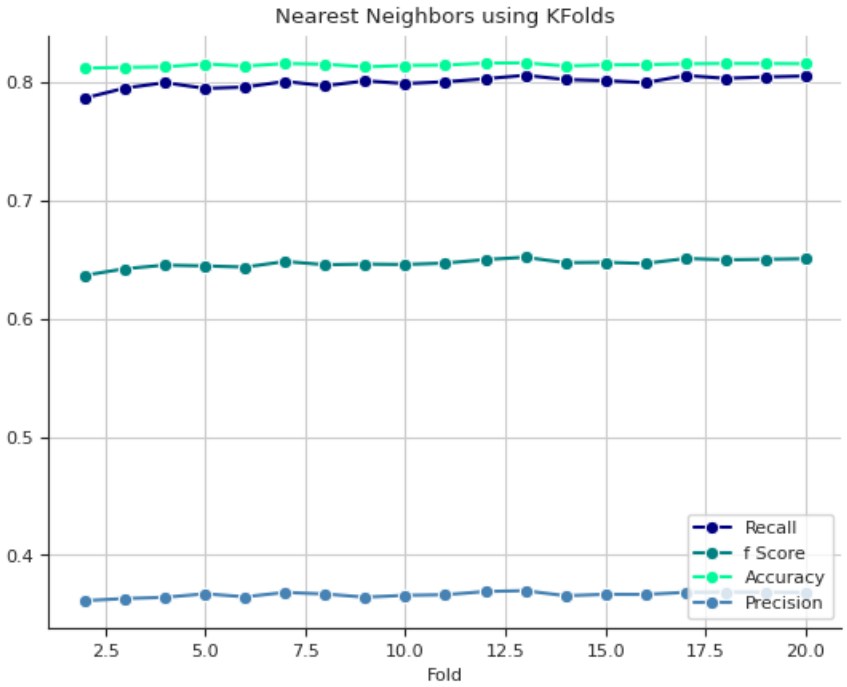

Figura 10 - Gráfico de performance do k-Nearest Neighbor com validação cruzada utilizando k-Fold

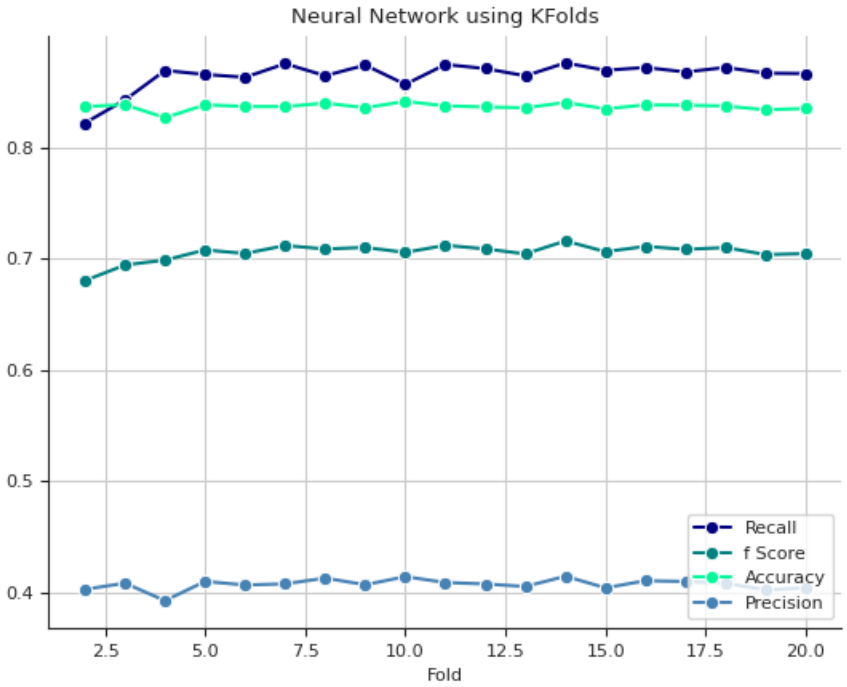

Figura 11 - Gráfico de performance da Rede Neural com validação cruzada utilizando k-Fold

Pela análise dos resultados, concluímos que o melhor valor de k para cada modelo:

- kNN: 4
- Árvore de Decisão: 7
- Rede Neural: 7

7. Validação dos Modelos

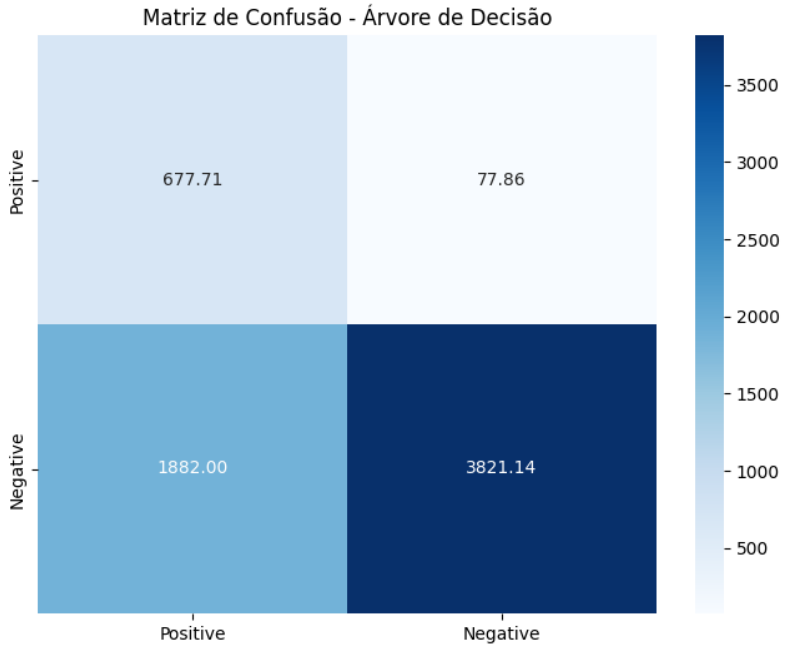

Figura 12 - Matriz de confusão da Árvore de Decisão construída a partir da média dos parâmetros (TP, FP, FN, TN) de cada fold sendo k = 4

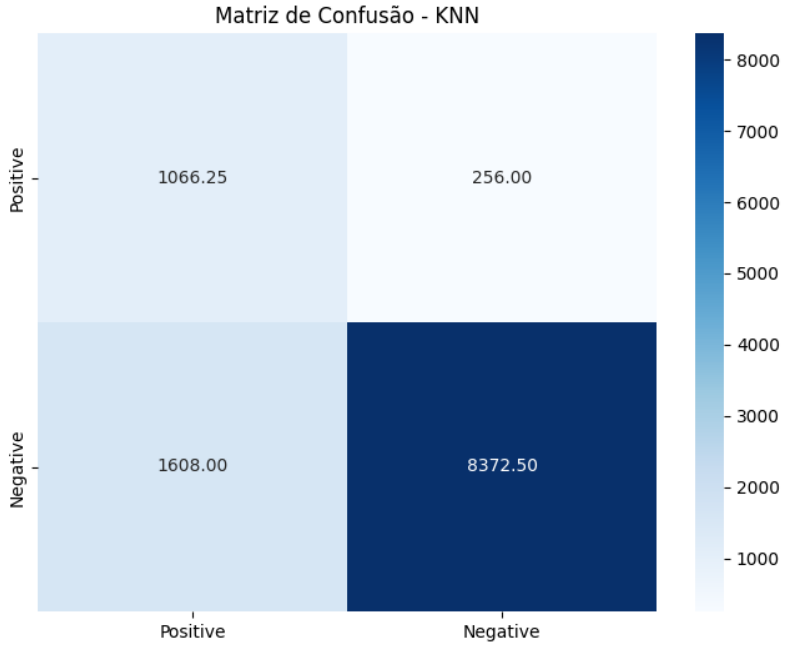

Figura 13 - Matriz de confusão do k-Nearest Neighbors construída a partir da média dos parâmetros (TP, FP, FN, TN) de cada fold sendo k = 7

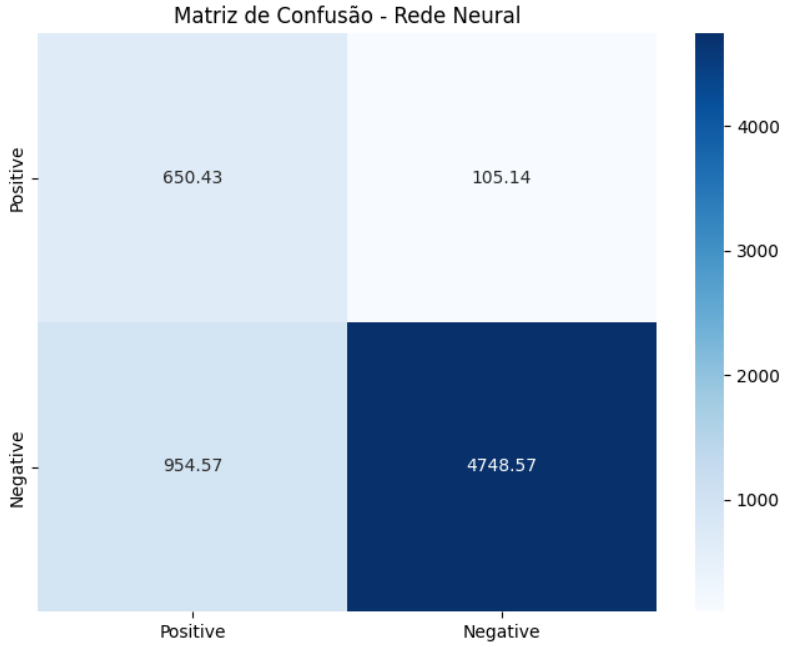

Figura 14 - Matriz de confusão da Rede Neural construída a partir da média dos parâmetros (TP, FP, FN, TN) de cada fold sendo k = 7

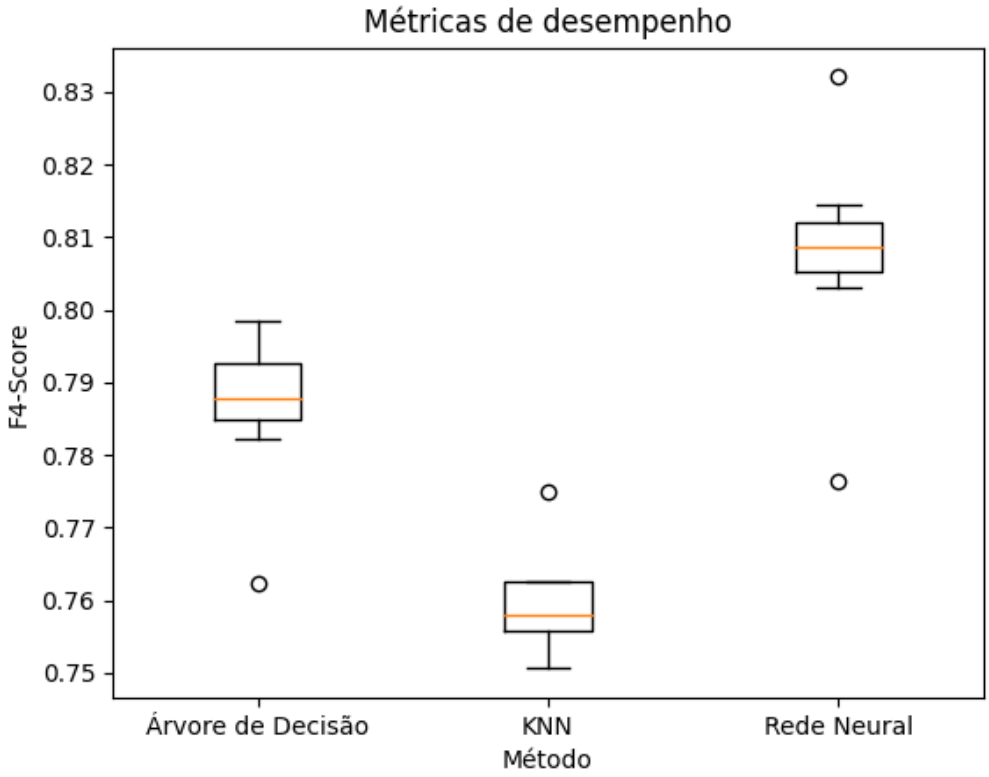

Figura 15 - Boxplot da métrica F4-Score dos três modelos treinados

8. Conclusão

Considerando o parâmetro F4-Score exibido na figura 15, pode-se concluir que a Rede Neural se mostrou melhor para processamento deste *dataset* em comparação aos outros modelos, apesar de apresentar maior variância.

Porém, é importante destacar que em termos de velocidade de treinamento/teste e desempenho, a Rede Neural deixou a desejar por possuir 3 camadas ocultas, demandando maior tempo de processamento até exibir seus resultados. No quesito velocidade de processamento, o modelo de Árvore de Decisão se destacou, seguido do modelo de k-Nearest Neighbors.