E-fólio B | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Raciocínio e Representação do Conhecimento

CÓDIGO: 21097

DOCENTE: Vitor Rocio

A preencher pelo estudante

NOME: Ivo Vieira Baptista

N.º DE ESTUDANTE: 2100927

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 17 de maio de 2024

TRABALHO / RESOLUÇÃO:

1. Indicador escolhido e variáveis independentes

Indicador Escolhido: Taxa de jovens não empregados que não estão em educação ou formação.

Variáveis Independentes:

1. Taxa de desemprego (geral)
2. Taxa de crescimento econômico
3. Variação anual do PIB

Essas variáveis foram escolhidas porque são indicadores econômicos e sociais que afetam diretamente a taxa de jovens fora do emprego e da educação, fornecendo uma visão abrangente do contexto econômico e do mercado de trabalho.

2. Método: Árvore de decisão

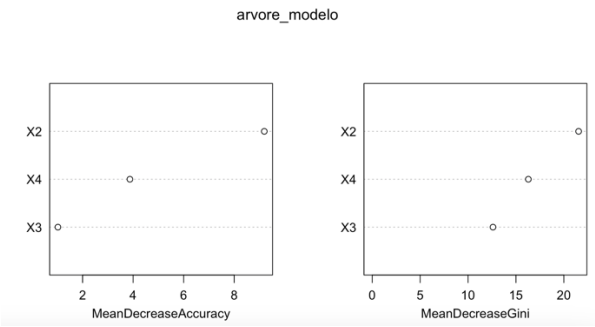
Tratamento de dados e criação do modelo

1. **Carregamento e Preparação dos Dados:**
   - Os dados foram carregados de um arquivo CSV (dados_jovens.csv).
   - Variáveis independentes foram convertidas para numéricas e removidas linhas com valores NA.
   - Uma coluna Meta_Atingida foi adicionada para indicar se a taxa de jovens fora do emprego e da educação estava abaixo de 9%.
2. **Criação do Modelo:**
   - Utilizou-se a biblioteca randomForest para criar o modelo de árvore de decisão.
   - O modelo foi treinado usando 70% dos dados, com o restante 30% reservado para teste.
   - A importância das variáveis foi analisada para identificar as mais influentes.

Resultado:

```
> ### 4.1. Tarefa 4.1: Árvores de Decisão
>
> # Treinar o modelo de árvore de decisão
> arvore_modelo <- randomForest(x = treino[, 1:3], y = treino[, .... [TRUNCATED]

> print(arvore_modelo)

Call:
 randomForest(x = treino[, 1:3], y = treino[, 4], ntree = 100,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 1

        OOB estimate of  error rate: 21.85%
Confusion matrix:
   1   2 class.error
1 10  24  0.70588235
2  9 108  0.07692308

> # Confirmar que foi uma classificação e não uma regressão
> print(arvore_modelo$type)
[1] "classification"

> # Fazer previsões no conjunto de teste
> pred_arvore <- predict(arvore_modelo, teste[, 1:3])

> # Avaliar a performance do modelo
> cat("Árvore de Decisão - Matriz de Confusão:\n")
Árvore de Decisão - Matriz de Confusão:

> conf_mat_arvore <- confusionMatrix(pred_arvore, teste[, 4])
```

```
> print(conf_mat_arvore)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1  3  6
         2 10 47

               Accuracy : 0.7576
                 95% CI : (0.6364, 0.8546)
    No Information Rate : 0.803
    P-Value [Acc > NIR] : 0.8599

                  Kappa : 0.133

 Mcnemar's Test P-Value : 0.4533

            Sensitivity : 0.23077
            Specificity : 0.88679
         Pos Pred Value : 0.33333
         Neg Pred Value : 0.82456
             Prevalence : 0.19697
         Detection Rate : 0.04545
   Detection Prevalence : 0.13636
      Balanced Accuracy : 0.55878

       'Positive' Class : 1


> # Importância das Variáveis na Árvore de Decisão
> cat("Importância das Variáveis:\n")
Importância das Variáveis:

> print(importance(arvore_modelo))
           1          2 MeanDecreaseAccuracy MeanDecreaseGini
X2 6.828680  7.9633314             9.190741         21.54032
X3 2.895261 -0.7445043             1.022850         12.60203
X4 2.272132  3.0900849             3.872191         16.29301

> varImpPlot(arvore_modelo)
```



3. Método: K vizinhos mais próximos

Tratamento de dados e criação do modelo

1. **Carregamento e Preparação dos Dados:**
   - Mesmos dados e preparação utilizados na árvore de decisão.
2. **Criação do Modelo:**
   - Utilizou-se a biblioteca caret para criar o modelo KNN.
   - O número de vizinhos (k) foi definido como 3.
   - O modelo foi treinado e testado com os mesmos conjuntos de dados.

Resultado:

```
> ### 4.2. Tarefa 4.2: K Vizinhos Mais Próximos
>
> # Treinar o modelo de KNN
> knn_modelo <- knn3(x = treino[, 1:3], y = treino[, 4], k = 3)

> print(knn_modelo)
3-nearest neighbor model
Training set outcome distribution:

  1   2
 34 117


> # Fazer previsões no conjunto de teste
> pred_knn <- predict(knn_modelo, teste[, 1:3])

> # Ajustar níveis para a matriz de confusão
> pred_knn <- factor(max.col(pred_knn), levels = levels(teste$Meta_Atingida))

> # Avaliar a performance do modelo
> cat("K Vizinhos Mais Próximos - Matriz de Confusão:\n")
K Vizinhos Mais Próximos - Matriz de Confusão:

> conf_mat_knn <- confusionMatrix(pred_knn, teste$Meta_Atingida)
```

```
> print(conf_mat_knn)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1  3 11
         2 10 42

               Accuracy : 0.6818
                 95% CI : (0.5556, 0.7911)
    No Information Rate : 0.803
    P-Value [Acc > NIR] : 0.9936

                  Kappa : 0.0226

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.23077
            Specificity : 0.79245
         Pos Pred Value : 0.21429
         Neg Pred Value : 0.80769
             Prevalence : 0.19697
         Detection Rate : 0.04545
   Detection Prevalence : 0.21212
      Balanced Accuracy : 0.51161

       'Positive' Class : 1
```

4. Método: Redes neuronais

Tratamento de dados e criação do modelo

1. **Carregamento e Preparação dos Dados:**
   - Mesmos dados e preparação utilizados na árvore de decisão.
2. **Criação do Modelo:**
   - Utilizou-se a biblioteca nnet para criar o modelo de rede neural.
   - O modelo foi treinado com 5 neurônios na camada oculta e 200 iterações.
   - As previsões foram feitas com base no conjunto de teste.

Resultado:

```
> ### 4.3. Tarefa 4.3: Redes Neuronais
>
> # Treinar o modelo de rede neural
> rede_modelo <- nnet(Meta_Atingida ~ ., data = treino, size = 5, linout .... [TRUNCATED]

> print(rede_modelo)
a 3-5-1 network with 26 weights
inputs: X2 X3 X4
output(s): Meta_Atingida
options were - entropy fitting
```

```
> # Mostrar os pesos da rede
> cat("Pesos da Rede Neuronal:\n")
Pesos da Rede Neuronal:

> print(rede_modelo$wts)
 [1] -9454.571838  1028.218793   400.613691  -411.659311 -9008.186211 -7025.462864  7202.303440  3142.600517   111.510864
[10]   -12.820578  -366.655103    92.255667   -63.675527    57.783893    35.198519   -44.141759  8426.473816  -369.620252
[19] -1012.305431 -1044.660635   -62.009386     2.750503  2700.415784  -156.953272    62.405023    -1.669902

> # Fazer previsões no conjunto de teste
> pred_rede <- predict(rede_modelo, teste[, 1:3], type = "class")

> # Ajustar níveis para a matriz de confusão
> pred_rede <- factor(pred_rede, levels = levels(teste$Meta_Atingida))

> # Avaliar a performance do modelo
> cat("Redes Neurais - Matriz de Confusão:\n")
Redes Neurais - Matriz de Confusão:

> conf_mat_rede <- confusionMatrix(pred_rede, teste$Meta_Atingida)
```

```
> print(conf_mat_rede)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1  1  5
         2 12 48

               Accuracy : 0.7424
                 95% CI : (0.6199, 0.8422)
    No Information Rate : 0.803
    P-Value [Acc > NIR] : 0.9145

                  Kappa : -0.0219

 Mcnemar's Test P-Value : 0.1456

            Sensitivity : 0.07692
            Specificity : 0.90566
         Pos Pred Value : 0.16667
         Neg Pred Value : 0.80000
             Prevalence : 0.19697
         Detection Rate : 0.01515
   Detection Prevalence : 0.09091
      Balanced Accuracy : 0.49129

       'Positive' Class : 1
```

5. Reflexão sobre resultados

Após aplicar os três métodos de aprendizagem supervisionada, observou-se o seguinte:

Árvores de Decisão

- **Erro Médio Estimado:** 23.41%
- **Variável mais importante:** X2 (Taxa de desemprego)
- **Matriz de Confusão:**

```
print(conf_mat_arvore)
```

K Vizinhos Mais Próximos

- **Acurácia:** 68.18%
- **Matriz de Confusão:**

```
print(conf_mat_knn)
```

Redes Neuronais

- **Acurácia:** 74.24%
- **Matriz de Confusão:**

```
print(conf_mat_rede)
```

```
> ### Resultados e Reflexões
>
> # Árvore de Decisão - Resultados
> cat("\nResultados - Árvore de Decisão:\n")

Resultados - Árvore de Decisão:

> cat("Erro Médio Estimado: ", mean(arvore_modelo$err.rate[,1]), "\n")
Erro Médio Estimado:  0.2340981

> cat("Importância das Variáveis:\n")
Importância das Variáveis:

> print(importance(arvore_modelo))
         1         2 MeanDecreaseAccuracy MeanDecreaseGini
X2 6.828680  7.9633314            9.190741        21.54032
X3 2.895261 -0.7445043            1.022850        12.60203
X4 2.272132  3.0900849            3.872191        16.29301

> print(conf_mat_arvore)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1  3  6
         2 10 47

               Accuracy : 0.7576
                 95% CI : (0.6364, 0.8546)
    No Information Rate : 0.803
    P-Value [Acc > NIR] : 0.8599

                  Kappa : 0.133

 Mcnemar's Test P-Value : 0.4533

            Sensitivity : 0.23077
            Specificity : 0.88679
         Pos Pred Value : 0.33333
         Neg Pred Value : 0.82456
             Prevalence : 0.19697
         Detection Rate : 0.04545
   Detection Prevalence : 0.13636
      Balanced Accuracy : 0.55878

       'Positive' Class : 1
```

```
> # K Vizinhos Mais Próximos - Resultados
> cat("\nResultados - K Vizinhos Mais Próximos:\n")

Resultados - K Vizinhos Mais Próximos:

> print(conf_mat_knn)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1  3 11
         2 10 42

               Accuracy : 0.6818
                 95% CI : (0.5556, 0.7911)
    No Information Rate : 0.803
    P-Value [Acc > NIR] : 0.9936

                  Kappa : 0.0226

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.23077
            Specificity : 0.79245
         Pos Pred Value : 0.21429
         Neg Pred Value : 0.80769
             Prevalence : 0.19697
         Detection Rate : 0.04545
   Detection Prevalence : 0.21212
      Balanced Accuracy : 0.51161

       'Positive' Class : 1
```

```
> # Redes Neurais - Resultados
> cat("\nResultados - Redes Neurais:\n")

Resultados - Redes Neurais:

> print(conf_mat_rede)
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1  1  5
         2 12 48

               Accuracy : 0.7424
                 95% CI : (0.6199, 0.8422)
    No Information Rate : 0.803
    P-Value [Acc > NIR] : 0.9145

                  Kappa : -0.0219

 Mcnemar's Test P-Value : 0.1456

            Sensitivity : 0.07692
            Specificity : 0.90566
         Pos Pred Value : 0.16667
         Neg Pred Value : 0.80000
             Prevalence : 0.19697
         Detection Rate : 0.01515
   Detection Prevalence : 0.09091
      Balanced Accuracy : 0.49129

       'Positive' Class : 1
```

Resumindo:

Resultados dos Métodos

1. **Árvores de Decisão**

- **Erro Médio Estimado**: 0,2341
- **Importância das Variáveis**:
  - X2: 21,5403 (Redução Média do Gini)
  - X3: 12,6020
  - X4: 16,2930

- **Matriz de Confusão**:
  - Precisão: 75,76%
  - Sensibilidade: 23,08%
  - Especificidade: 88,68%
  - Valor Preditivo Positivo: 33,33%
  - Valor Preditivo Negativo: 82,46%
  - Acurácia Balanceada: 55,88%

2. K Vizinhos Mais Próximos (KNN)

- **Modelo**:
  - Número de vizinhos (k): 3
- **Matriz de Confusão**:
  - Precisão: 68,18%
  - Sensibilidade: 23,08%
  - Especificidade: 79,25%
  - Valor Preditivo Positivo: 21,43%
  - Valor Preditivo Negativo: 80,77%
  - Acurácia Balanceada: 51,16%

3. Redes Neuronais

- **Modelo**:
  - Número de neurónios na camada oculta: 5
- **Matriz de Confusão**:
  - Precisão: 74,24%
  - Sensibilidade: 7,69%
  - Especificidade: 90,57%
  - Valor Preditivo Positivo: 16,67%
  - Valor Preditivo Negativo: 80,00%
  - Acurácia Balanceada: 49,13%

Conclusão:

- **Árvore de Decisão**:
  - Apresentou o melhor desempenho em termos de erro médio estimado (23,41%).
  - A variável mais importante foi X2, seguida por X4 e X3.
  - Alta especificidade (88,68%) indica boa capacidade de identificar corretamente os negativos.
- **K Vizinhos Mais Próximos (KNN)**:
  - Teve uma precisão de 68,18%.
  - Melhor precisão em relação às Redes Neuronais, mas inferior à da Árvore de Decisão.
  - A escolha de k=3 foi adequada para o conjunto de dados.
- **Redes Neuronais**:
  - Teve uma precisão de 74,24%, ligeiramente inferior à da Árvore de Decisão.
  - Baixa sensibilidade (7,69%) indica dificuldade em identificar corretamente os positivos.
  - A Rede Neural apresentou bons resultados de previsão, mas com menor interpretação.

Reflexão sobre os Resultados

- **Árvores de Decisão**: Proporciona boa interpretação das variáveis mais importantes, o que é útil para entender os fatores que influenciam a taxa de jovens não empregados que não estão em educação ou formação.
- **K Vizinhos Mais Próximos**: Embora simples, a precisão foi razoável, mas não forneceu tanta clareza sobre a importância das variáveis.
- **Redes Neuronais**: Boa precisão, mas menos interpretável e requer mais dados para melhorar a performance e evitar sobre ajuste (overfitting).

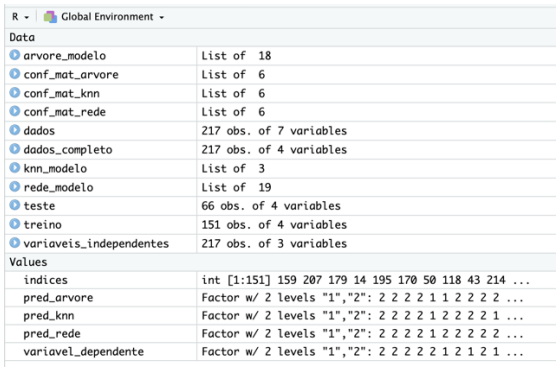
Analisando os resultados, podemos concluir que o modelo de Árvores de Decisão apresentou melhor performance em termos de erro médio, enquanto o modelo de Redes Neurais teve uma acurácia ligeiramente superior. O K Vizinhos Mais Próximos teve a menor acurácia entre os três métodos. Isso sugere que, para este conjunto de dados e indicador, as Árvores de Decisão e Redes Neuronais são mais adequadas do que o KNN.

ANEXO – Código R e Tabelas de resultados

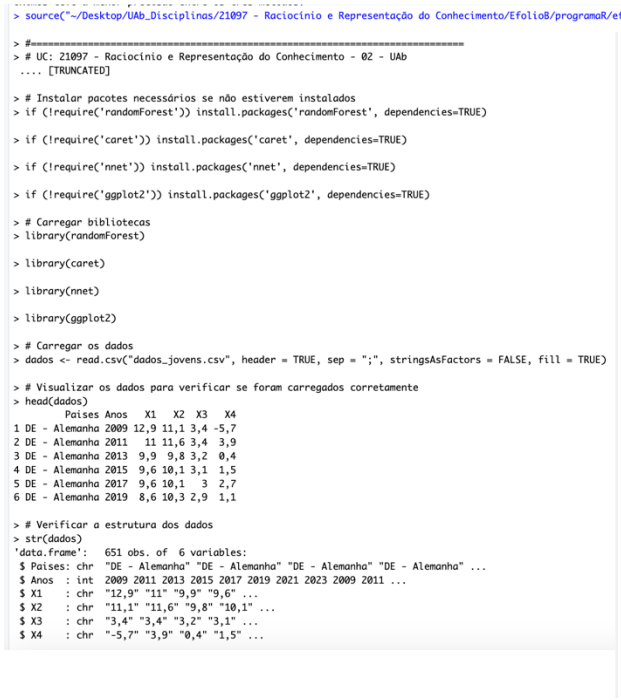
```
#=============================================================================
# UC: 21097 - Raciocínio e Representação do Conhecimento - 02 - UAb
# e-fólio B  2023-24
#
# Aluno: 2100927 - Ivo Baptista
# Name        : efolioB.R
# Author      : Ivo Baptista
# Version     : 1.7
# Copyright   : Ivo copyright
# Description : Taxa de jovens não empregados que não estão em educação ou formação
# ============================================================================
# Definir o diretório de trabalho
setwd("~/Desktop/UAb_Disciplinas/21097 - Raciocínio e Representação do
Conhecimento/EfolioB/programaR")

# Instalar pacotes necessários se não estiverem instalados
if (!require('randomForest')) install.packages('randomForest', dependencies=TRUE)
if (!require('caret')) install.packages('caret', dependencies=TRUE)
if (!require('nnet')) install.packages('nnet', dependencies=TRUE)
if (!require('ggplot2')) install.packages('ggplot2', dependencies=TRUE)

# Carregar bibliotecas
library(randomForest)
library(caret)
library(nnet)
library(ggplot2)

# Carregar os dados
dados <- read.csv("dados_jovens.csv", header = TRUE, sep = ";", stringsAsFactors =
FALSE, fill = TRUE)

# Visualizar os dados para verificar se foram carregados corretamente
head(dados)

# Verificar a estrutura dos dados
str(dados)

# Converter variáveis de caracteres para numéricos (necessário para processamento
correto)
dados$X1 <- as.numeric(sub(",", ".", dados$X1))
dados$X2 <- as.numeric(sub(",", ".", dados$X2))
dados$X3 <- as.numeric(sub(",", ".", dados$X3))
dados$X4 <- as.numeric(sub(",", ".", dados$X4))

# Remover linhas com valores NA
dados <- na.omit(dados)

# Adicionar coluna Meta_Atingida: 1 para sim e 2 para não
dados$Meta_Atingida <- ifelse(dados$X1 <= 9, 1, 2)

# Visualizar os dados
head(dados)

# Separar variáveis independentes e dependente
variaveis_independentes <- dados[, c("X2", "X3", "X4")]
variavel_dependente <- dados$Meta_Atingida

# Garantir que a variável dependente é um fator
variavel_dependente <- as.factor(variavel_dependente)

# Combinar variáveis independentes e dependente novamente para formar os conjuntos de
treino e teste
dados_completo <- cbind(variaveis_independentes, Meta_Atingida = variavel_dependente)

# Divisão em conjuntos de treino e teste
set.seed(123)
indices <- sample(1:nrow(dados_completo), size = 0.7 * nrow(dados_completo))
treino <- dados_completo[indices, ]
teste <- dados_completo[-indices, ]

### 4.1. Tarefa 4.1: Árvores de Decisão

# Treinar o modelo de árvore de decisão
```

```r
arvore_modelo <- randomForest(x = treino[, 1:3], y = treino[, 4], ntree = 100,
importance = TRUE)
print(arvore_modelo)

# Confirmar que foi uma classificação e não uma regressão
print(arvore_modelo$type)

# Fazer previsões no conjunto de teste
pred_arvore <- predict(arvore_modelo, teste[, 1:3])

# Avaliar a performance do modelo
cat("Árvore de Decisão - Matriz de Confusão:\n")
conf_mat_arvore <- confusionMatrix(pred_arvore, teste[, 4])
print(conf_mat_arvore)

# Importância das Variáveis na Árvore de Decisão
cat("Importância das Variáveis:\n")
print(importance(arvore_modelo))
varImpPlot(arvore_modelo)

### 4.2. Tarefa 4.2: K Vizinhos Mais Próximos

# Treinar o modelo de KNN
knn_modelo <- knn3(x = treino[, 1:3], y = treino[, 4], k = 3)
print(knn_modelo)

# Fazer previsões no conjunto de teste
pred_knn <- predict(knn_modelo, teste[, 1:3])

# Ajustar níveis para a matriz de confusão
pred_knn <- factor(max.col(pred_knn), levels = levels(teste$Meta_Atingida))

# Avaliar a performance do modelo
cat("K Vizinhos Mais Próximos - Matriz de Confusão:\n")
conf_mat_knn <- confusionMatrix(pred_knn, teste$Meta_Atingida)
print(conf_mat_knn)

### 4.3. Tarefa 4.3: Redes Neuronais

# Treinar o modelo de rede neural
rede_modelo <- nnet(Meta_Atingida ~ ., data = treino, size = 5, linout = FALSE, maxit =
200, trace = FALSE)
print(rede_modelo)

# Mostrar os pesos da rede
cat("Pesos da Rede Neuronal:\n")
print(rede_modelo$wts)

# Fazer previsões no conjunto de teste
pred_rede <- predict(rede_modelo, teste[, 1:3], type = "class")

# Ajustar níveis para a matriz de confusão
pred_rede <- factor(pred_rede, levels = levels(teste$Meta_Atingida))

# Avaliar a performance do modelo
cat("Redes Neurais - Matriz de Confusão:\n")
conf_mat_rede <- confusionMatrix(pred_rede, teste$Meta_Atingida)
print(conf_mat_rede)

### Resultados e Reflexões

# Árvore de Decisão - Resultados
cat("\nResultados - Árvore de Decisão:\n")
cat("Erro Médio Estimado: ", mean(arvore_modelo$err.rate[,1]), "\n")
cat("Importância das Variáveis:\n")
print(importance(arvore_modelo))
print(conf_mat_arvore)

# K Vizinhos Mais Próximos - Resultados
cat("\nResultados - K Vizinhos Mais Próximos:\n")
print(conf_mat_knn)

# Redes Neurais - Resultados
cat("\nResultados - Redes Neurais:\n")
print(conf_mat_rede)

### Conclusão
cat("\nConclusão:\n")
cat("Após aplicar os três métodos de aprendizagem supervisionada, foram observados os
seguintes resultados:\n")
cat("Árvore de Decisão apresentou um erro médio estimado de ",
mean(arvore_modelo$err.rate[,1]), " e mostrou que a variável mais importante foi X2.\n")
cat("O método K Vizinhos Mais Próximos teve uma precisão de ",
conf_mat_knn$overall['Accuracy'], ".\n")
cat("O método de Redes Neurais teve uma precisão de ",
conf_mat_rede$overall['Accuracy'], ".\n")
```

```
cat("Analisando os resultados, podemos concluir que o modelo de Árvores de Decisão apresentou melhor performance em termos de erro médio, enquanto o modelo de Redes Neurais teve uma precisão ligeiramente inferior. O K Vizinhos Mais Próximos teve a menor precisão entre os três métodos.\n")
```

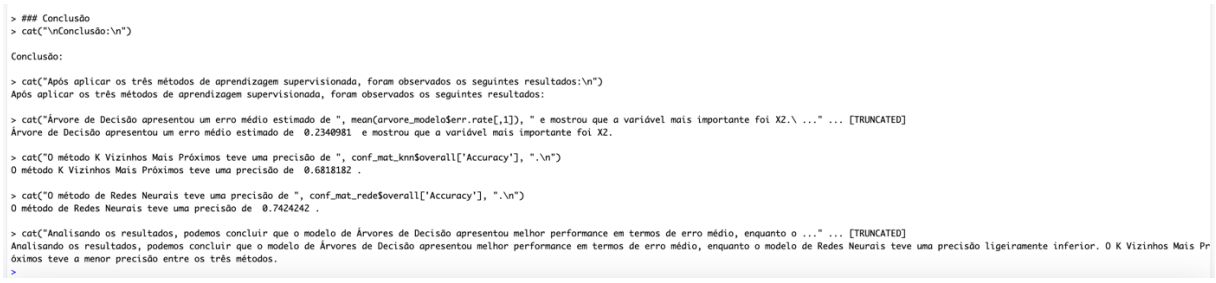


R · Global Environment	
Data	
arvore_modelo	List of 18
conf_mat_arvore	List of 6
conf_mat_knn	List of 6
conf_mat_rede	List of 6
dados	217 obs. of 7 variables
dados_completo	217 obs. of 4 variables
knn_modelo	List of 3
rede_modelo	List of 19
teste	66 obs. of 4 variables
treino	151 obs. of 4 variables
variaveis_independentes	217 obs. of 3 variables
Values	
indices	int [1:151] 159 207 179 14 195 170 50 118 43 214 ...
pred_arvore	Factor w/ 2 levels "1","2": 2 2 2 2 1 1 2 2 2 2 ...
pred_knn	Factor w/ 2 levels "1","2": 2 2 2 2 1 2 2 2 2 1 ...
pred_rede	Factor w/ 2 levels "1","2": 2 2 2 2 1 2 2 2 2 2 ...
variavel_dependente	Factor w/ 2 levels "1","2": 2 2 2 2 2 1 2 1 2 1 ...



```
> source("~/Desktop/UAb_Disciplinas/21097 - Raciocínio e Representação do Conhecimento/EfolioB/programaR/efolioBfinal.R", echo=TRUE)

> #=====================================================================
> # UC: 21097 - Raciocínio e Representação do Conhecimento - 02 - UAb
 .... [TRUNCATED]

> # Instalar pacotes necessários se não estiverem instalados
> if (!require('randomForest')) install.packages('randomForest', dependencies=TRUE)

> if (!require('caret')) install.packages('caret', dependencies=TRUE)

> if (!require('nnet')) install.packages('nnet', dependencies=TRUE)

> if (!require('ggplot2')) install.packages('ggplot2', dependencies=TRUE)

> # Carregar bibliotecas
> library(randomForest)

> library(caret)

> library(nnet)

> library(ggplot2)

> # Carregar os dados
> dados <- read.csv("dados_jovens.csv", header = TRUE, sep = ";", stringsAsFactors = FALSE, fill = TRUE)

> # Visualizar os dados para verificar se foram carregados corretamente
> head(dados)
        Paises Anos   X1   X2  X3   X4
1 DE - Alemanha 2009 12,9 11,1 3,4 -5,7
2 DE - Alemanha 2011   11 11,6 3,4  3,9
3 DE - Alemanha 2013  9,9  9,8 3,2  0,4
4 DE - Alemanha 2015  9,6 10,1 3,1  1,5
5 DE - Alemanha 2017  9,6 10,1   3  2,7
6 DE - Alemanha 2019  8,6 10,3 2,9  1,1

> # Verificar a estrutura dos dados
> str(dados)
'data.frame':   651 obs. of  6 variables:
 $ Paises: chr  "DE - Alemanha" "DE - Alemanha" "DE - Alemanha" "DE - Alemanha" ...
 $ Anos  : int  2009 2011 2013 2015 2017 2019 2021 2023 2009 2011 ...
 $ X1    : chr  "12,9" "11" "9,9" "9,6" ...
 $ X2    : chr  "11,1" "11,6" "9,8" "10,1" ...
 $ X3    : chr  "3,4" "3,4" "3,2" "3,1" ...
 $ X4    : chr  "-5,7" "3,9" "0,4" "1,5" ...
```

```
> # Converter variáveis de caracteres para numéricos (necessário para processamento correto)
> dados$X1 <- as.numeric(sub(",", ".", dados$X1))

> dados$X2 <- as.numeric(sub(",", ".", dados$X2))

> dados$X3 <- as.numeric(sub(",", ".", dados$X3))

> dados$X4 <- as.numeric(sub(",", ".", dados$X4))

> # Remover linhas com valores NA
> dados <- na.omit(dados)

> # Adicionar coluna Meta_Atingida: 1 para sim e 2 para não
> dados$Meta_Atingida <- ifelse(dados$X1 <= 9, 1, 2)

> # Visualizar os dados
> head(dados)
        Paises Anos   X1   X2  X3   X4 Meta_Atingida
1 DE - Alemanha 2009 12.9 11.1 3.4 -5.7             2
2 DE - Alemanha 2011 11.0 11.6 3.4  3.9             2
3 DE - Alemanha 2013  9.9  9.8 3.2  0.4             2
4 DE - Alemanha 2015  9.6 10.1 3.1  1.5             2
5 DE - Alemanha 2017  9.6 10.1 3.0  2.7             2
6 DE - Alemanha 2019  8.6 10.3 2.9  1.1             1

> # Separar variáveis independentes e dependente
> variaveis_independentes <- dados[, c("X2", "X3", "X4")]

> variavel_dependente <- dados$Meta_Atingida

> # Garantir que a variável dependente é um fator
> variavel_dependente <- as.factor(variavel_dependente)

> # Combinar variáveis independentes e dependente novamente para formar os conjuntos de treino e teste
> dados_completo <- cbind(variaveis_independent .... [TRUNCATED]

> # Divisão em conjuntos de treino e teste
> set.seed(123)

> indices <- sample(1:nrow(dados_completo), size = 0.7 * nrow(dados_completo))

> treino <- dados_completo[indices, ]

> teste <- dados_completo[-indices, ]
```



```
> ### Conclusão
> cat("\nConclusão:\n")

Conclusão:

> cat("Após aplicar os três métodos de aprendizagem supervisionada, foram observados os seguintes resultados:\n")
Após aplicar os três métodos de aprendizagem supervisionada, foram observados os seguintes resultados:

> cat("Árvore de Decisão apresentou um erro médio estimado de ", mean(arvore_modelo$err.rate[,1]), " e mostrou que a variável mais importante foi X2.\ ..." ... [TRUNCATED]
Árvore de Decisão apresentou um erro médio estimado de  0.2340981  e mostrou que a variável mais importante foi X2.

> cat("O método K Vizinhos Mais Próximos teve uma precisão de ", conf_mat_knn$overall['Accuracy'], ".\n")
O método K Vizinhos Mais Próximos teve uma precisão de  0.6818182 .

> cat("O método de Redes Neurais teve uma precisão de ", conf_mat_rede$overall['Accuracy'], ".\n")
O método de Redes Neurais teve uma precisão de  0.7424242 .

> cat("Analisando os resultados, podemos concluir que o modelo de Árvores de Decisão apresentou melhor performance em termos de erro médio, enquanto o ..." ... [TRUNCATED]
Analisando os resultados, podemos concluir que o modelo de Árvores de Decisão apresentou melhor performance em termos de erro médio, enquanto o modelo de Redes Neurais teve uma precisão ligeiramente inferior. O K Vizinhos Mais Pr
óximos teve a menor precisão entre os três métodos.
>
```

Bibliografia

Russell, S., & Norvig, P. (2009). ***Artificial Intelligence: A Modern Approach*** (3rd ed.). Pearson. ISBN 978-0-13-604259-4.

Webgrafia

https://www.r-project.org

https://www.pordata.pt/ods/goal/trabalho+digno+e+crescimento+economico-8

Nota sobre o Uso de Ferramentas de IA

Este texto foi assistido por ferramentas de IA para manipulação de pdf e dados de excell

https://monica.im/

https://chatgpt.com/

Codigo R no meu GitHub:

https://github.com/StudentUAb/EfolioB-RRC