

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Disciplina: Ciência de Dados

Curso: Sistemas de Informação

Professora: Elaine Ribeiro Faria

Trabalho Final – Comparação de Classificadores

Estudantes:

Nome	Matrícula
Paulo Daniel Forti da Fonseca	12311BSI321

Tema: Comparação de classificadores supervisionados em duas bases reais distintas.

Sumário

1	Introdução e Objetivos	3
2	Bases de Dados e Exploração	4
2.1	Base 1: Adult Census Income	4
2.1.1	Descrição geral	4
2.1.2	Atributos, tipos e valores ausentes	4
2.1.3	Exploração e observações	5
2.2	Base 2: Bank Marketing	5
2.2.1	Descrição geral	5
2.2.2	Atributos, tipos e valores ausentes	6
2.2.3	Exploração e observações	6
3	Metodologia	8
3.1	Ferramentas e ambiente	8
3.2	Pré-processamento	8
3.3	Algoritmos de classificação	9
3.4	Estratégia de divisão treino/teste (Nested Stratified K-Fold)	10
3.5	Métricas de avaliação	12
3.6	Parametrizações testadas (por algoritmo)	14
4	Resultados e Discussão	16
4.1	Tabela-síntese dos resultados (média \pm desvio padrão)	16
4.2	Discussão por base de dados	16
4.2.1	Adult Census Income	16
4.2.2	Bank Marketing	17
4.3	Comparação geral: houve um algoritmo melhor nas duas bases?	17
5	Uso de LLMs	18
5.1	Onde utilizamos	18
5.2	Prompts utilizados (exemplos)	18

Capítulo 1

Introdução e Objetivos

Este trabalho tem como objetivo aplicar, avaliar e interpretar o desempenho de algoritmos de classificação supervisionada em duas bases de dados reais distintas, enfatizando: (i) entendimento/exploração dos dados, (ii) pré-processamento adequado, (iii) comparação crítica entre algoritmos, e (iv) avaliação com múltiplas métricas e diferentes parametrizações.

Objetivos específicos:

- Selecionar duas bases públicas reais e descrever seus atributos, tipos de dados, classe-alvo e presença de valores ausentes;
- Executar um pipeline completo de pré-processamento para suportar modelos baseados em distância (KNN) e modelos baseados em árvores;
- Comparar três algoritmos (KNN, Árvore de Decisão e Random Forest), testando diferentes parametrizações;
- Adotar uma estratégia de divisão treino/teste robusta (Nested Stratified K-Fold) para estimar desempenho generalizável;
- Avaliar os modelos por múltiplas métricas (Accuracy, F1 e ROC-AUC) e discutir criticamente os resultados.

Capítulo 2

Bases de Dados e Exploração

2.1 Base 1: Adult Census Income

Fonte: Kaggle.

Link: <https://www.kaggle.com/datasets/uciml/adult-census-income>

2.1.1 Descrição geral

A base Adult Census Income contém **32.561 instâncias** e **15 atributos**. O problema é de classificação binária, em que a classe (`income`) indica se a renda anual é $\leq 50K$ ou $>50K$. A distribuição observada foi:

- $\leq 50K$: 24.720 instâncias (aprox. 75,9%)
- $>50K$: 7.841 instâncias (aprox. 24,1%)

Assim, há **desbalanceamento moderado** (classe positiva minoritária).

2.1.2 Atributos, tipos e valores ausentes

Classe (alvo): `income` (categórico: $\leq 50K$ / $>50K$)

Foram identificados valores ausentes concentrados em três atributos categóricos, totalizando: `occupation` (1843), `workclass` (1836) e `native.country` (583). Os demais atributos não apresentaram ausentes.

Tabela 2.1: Atributos da base Adult Census Income: tipo e presença de ausentes.

Atributo	Tipo	Ausentes	Descrição (resumo)
age	Numérico	(intelectivo)	Não Idade do indivíduo.
workclass	Categórico	Sim (1836)	Tipo de ocupação/setor (ex.: privado, governo).

Atributo	Tipo		Ausentes	Descrição (resumo)
fnlwgt	Numérico	(in- teiro)	Não	Peso amostral do censo.
education	Categórico		Não	Nível educacional (texto).
education.num	Numérico	(in- teiro)	Não	Nível educacional (codificação numérica).
marital.status	Categórico		Não	Estado civil.
occupation	Categórico		Sim (1843)	Ocupação/profissão.
relationship	Categórico		Não	Relação familiar (ex.: husband, not-in-family).
race	Categórico		Não	Raça.
sex	Categórico		Não	Sexo.
capital.gain	Numérico	(in- teiro)	Não	Ganho de capital.
capital.loss	Numérico	(in- teiro)	Não	Perda de capital.
hours.per.week	Numérico	(in- teiro)	Não	Horas trabalhadas por semana.
native.country	Categórico		Sim (583)	País de origem.
income	Categórico (classe)		Não	Classe-alvo: <=50K/>50K.

2.1.3 Exploração e observações

As variáveis incluem uma mistura de atributos numéricos e categóricos; isso exige pré-processamento com codificação apropriada de categóricos e padronização dos numéricos, especialmente para o KNN, que é sensível à escala. Além disso, o desbalanceamento (classe positiva minoritária) motiva o uso de métricas além da acurácia, como F1 e ROC-AUC.

2.2 Base 2: Bank Marketing

Fonte: Kaggle.

Link: <https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset>

2.2.1 Descrição geral

A base Bank Marketing contém **11.162 instâncias** e **17 atributos**. O problema é de classificação binária, em que a classe (**deposit**) indica se o cliente realizou um depósito (**yes**) ou não (**no**). A distribuição observada foi:

- no: 5.873 instâncias (aprox. 52,6%)
- yes: 5.289 instâncias (aprox. 47,4%)

O dataset é **quase balanceado**.

2.2.2 Atributos, tipos e valores ausentes

Classe (alvo): deposit (categórico: no / yes)

Embora a base não apresente valores ausentes explícitos no formato tradicional, foram identificados valores categóricos codificados como `unknown`, os quais foram tratados neste trabalho como **ausentes**. Esses valores concentram-se nos atributos `poutcome` (8326), `contact` (2346), `education` (497) e `job` (70). Os demais atributos não apresentaram ausências segundo esse critério.

Tabela 2.2: Atributos da base Bank Marketing: tipo e presença de ausentes.

Atributo	Tipo	Ausentes	Descrição (resumo)
age	Numérico (inteiro)	(in- teiro)	Não Idade.
job	Categórico	Sim (70)	Ocupação.
marital	Categórico	Não	Estado civil.
education	Categórico	Sim (497)	Escolaridade.
default	Categórico	Não	Inadimplência (sim/não).
balance	Numérico (inteiro)	(in- teiro)	Não Saldo bancário.
housing	Categórico	Não	Possui financiamento imobiliário (sim/não).
loan	Categórico	Não	Possui empréstimo (sim/não).
contact	Categórico	Sim (2346)	Tipo de contato.
day	Numérico (inteiro)	(in- teiro)	Não Dia do mês do contato.
month	Categórico	Não	Mês do contato.
duration	Numérico (inteiro)	(in- teiro)	Não Duração do contato (segundos).
campaign	Numérico (inteiro)	(in- teiro)	Nº de contatos na campanha.
pdays	Numérico (inteiro)	(in- teiro)	Não Dias desde o último contato pré- vio.
previous	Numérico (inteiro)	(in- teiro)	Nº de contatos anteriores.
poutcome	Categórico	Sim (8326)	Resultado da campanha ante- rior.
deposit	Categórico (classe)	Não	Classe-alvo: no/yes.

2.2.3 Exploração e observações

A base mistura atributos numéricos e categóricos. Além disso, alguns atributos categóricos apresentam valores `unknown`, tratados neste trabalho como ausentes. Assim, o pré-processamento dessa base envolveu tratamento dessas ausências, codificação categórica por *One-Hot Encoding* e padronização dos atributos numéricos para o KNN.

Um ponto importante é que a maior concentração de ausências ocorre no atributo `poutcome`,

seguido por `contact`. Isso indica que, embora a base seja quase balanceada em relação à variável-alvo, ela possui ausências relevantes em parte dos atributos explicativos, o que torna o tratamento adequado desses valores uma etapa importante da metodologia.

Capítulo 3

Metodologia

3.1 Ferramentas e ambiente

Os experimentos foram realizados em **Python**, utilizando a biblioteca **scikit-learn** para pré-processamento, validação cruzada e treinamento dos modelos.

3.2 Pré-processamento

Como as bases possuem atributos numéricos e categóricos, foi adotado um pipeline de pré-processamento implementado com **ColumnTransformer** e **Pipeline**. Essa organização permitiu aplicar transformações diferentes para atributos numéricos e categóricos de forma integrada ao processo de validação cruzada, evitando vazamento de informação entre treino e teste.

O pipeline adotado aplica:

- **Padronização de ausências implícitas:** valores especiais utilizados para representar ausência, como ? e unknown, foram convertidos para valores ausentes reais antes da imputação;
- **Tratamento de ausentes:**
 - Numéricos: imputação por *mediana*, por ser uma estratégia mais robusta à presença de outliers;
 - Categóricos: imputação por *moda* (*most frequent*), preservando categorias válidas no conjunto de dados.
- **Codificação de categóricos:** **One-Hot Encoding**, pois os algoritmos utilizados não operam diretamente sobre atributos textuais;
- **Padronização de numéricos:** **StandardScaler** (média 0 e desvio padrão 1), importante principalmente para o KNN, que é sensível à escala das variáveis.

Na base Adult Census Income, os ausentes estavam concentrados em **workclass**, **occupation** e **native.country**. Já na base Bank Marketing, valores categóricos codificados como **unknown** foram tratados como ausentes, especialmente nos atributos **job**, **education**, **contact** e **poutcome**.

Em ambos os casos, optou-se por imputação para evitar perda de instâncias e preservar informação potencialmente útil para a classificação.

Observação metodológica: embora a categoria `unknown` possa, em alguns casos, carregar informação própria, neste trabalho ela foi tratada de forma uniforme como ausência implícita, permitindo um tratamento padronizado das duas bases dentro do pipeline experimental adotado.

3.3 Algoritmos de classificação

Foram utilizados três algoritmos:

- KNN (K-Nearest Neighbors);
- Árvore de Decisão;
- Random Forest.

Descrição do algoritmo adicional (Random Forest)

O **Random Forest** é um método de aprendizado de máquina do tipo **ensemble**, ou seja, ele combina vários modelos mais simples para formar um modelo final mais robusto. No caso, esses modelos base são **árvores de decisão**. A principal ideia do Random Forest é reduzir a alta variância de uma árvore individual, melhorando a capacidade de generalização do classificador em dados não vistos.

Uma árvore de decisão isolada pode se ajustar muito bem ao conjunto de treino, mas também pode sofrer de **overfitting**, isto é, aprender padrões muito específicos ou ruídos dos dados. O Random Forest tenta contornar esse problema construindo várias árvores diferentes e combinando suas respostas. Assim, em vez de depender de uma única árvore possivelmente instável, o modelo final se baseia na decisão conjunta de várias árvores.

Intuição de funcionamento: cada árvore da floresta é treinada com uma versão ligeiramente diferente do conjunto de dados e, além disso, analisa subconjuntos aleatórios de atributos ao longo do treinamento. Isso faz com que as árvores não sejam idênticas entre si. Como consequência, os erros cometidos por algumas árvores podem ser compensados pelas demais, tornando a predição final mais estável.

Passos do algoritmo:

1. Definir o número de árvores B que irão compor a floresta;
2. Para cada árvore $b = 1, 2, \dots, B$:
 - (a) gerar uma amostra *bootstrap* do conjunto de treino, isto é, uma amostra aleatória **com reposição**;
 - (b) treinar uma árvore de decisão nessa amostra;
 - (c) durante o crescimento da árvore, em cada nó, considerar apenas um **subconjunto aleatório de atributos** como candidatos para realizar a divisão;
3. Após treinar todas as árvores, para classificar uma nova instância, obter a predição de cada árvore individualmente;

4. Combinar as previsões por **votação majoritária** no caso de classificação.

Por que usar amostragem bootstrap? A amostragem *bootstrap* faz com que cada árvore seja treinada em um conjunto ligeiramente diferente, aumentando a diversidade entre as árvores. Essa diversidade é importante porque, se todas as árvores fossem treinadas exatamente nos mesmos dados da mesma forma, elas tenderiam a ser muito parecidas, reduzindo a vantagem do ensemble.

Por que usar subconjuntos aleatórios de atributos? Mesmo com *bootstrap*, ainda poderia acontecer de várias árvores se tornarem muito semelhantes, especialmente quando existem atributos muito preditivos. Ao limitar aleatoriamente quais atributos podem ser testados em cada divisão, o Random Forest aumenta ainda mais a diversidade entre as árvores e reduz a correlação entre elas.

Vantagens do Random Forest:

- geralmente apresenta melhor capacidade de generalização do que uma árvore de decisão isolada;
- reduz a variância do modelo;
- tende a ser menos suscetível a overfitting;
- lida bem com relações não lineares entre os atributos;
- costuma ter bom desempenho em bases tabulares heterogêneas.

Desvantagens do Random Forest:

- é menos interpretável do que uma única árvore de decisão;
- possui maior custo computacional para treinamento e predição;
- pode dificultar a explicação detalhada de uma decisão individual do modelo.

Relação com este trabalho: Neste trabalho, o Random Forest foi escolhido como o terceiro classificador por ser um algoritmo mais robusto e por complementar bem os outros dois modelos avaliados. Enquanto o KNN é baseado em distância e a Árvore de Decisão é um modelo único e mais instável, o Random Forest representa uma abordagem ensemble que busca combinar o poder preditivo de várias árvores para obter melhor desempenho e maior estabilidade.

3.4 Estratégia de divisão treino/teste (Nested Stratified K-Fold)

Para avaliar os modelos de forma mais confiável, foi utilizada uma estratégia de validação cruzada aninhada, chamada **Nested Stratified K-Fold**. Essa abordagem é mais robusta do que uma única divisão treino/teste (*holdout*), pois reduz a dependência de uma partição específica dos dados e separa corretamente o processo de **seleção de hiperparâmetros** do processo de **avaliação final** do modelo.

Antes de descrever a estratégia, é importante distinguir dois conceitos:

- **Parâmetros do modelo:** são valores aprendidos automaticamente durante o treinamento, como, por exemplo, as regras de divisão construídas em uma árvore;
- **Hiperparâmetros:** são configurações definidas antes do treinamento e que controlam o comportamento do algoritmo, como o número de vizinhos no KNN (`n_neighbors`), a profundidade máxima de uma árvore (`max_depth`) ou a quantidade de árvores no Random Forest (`n_estimators`).

O processo de escolher os melhores valores para esses hiperparâmetros é chamado de **hiperparametrização** ou **ajuste de hiperparâmetros**. Neste trabalho, essa etapa foi realizada por meio do `GridSearchCV`.

O que é GridSearchCV

O `GridSearchCV` é um método de busca exaustiva em grade com validação cruzada. Seu funcionamento consiste em definir previamente um conjunto de valores possíveis para os hiperparâmetros de um algoritmo e testar **todas as combinações** entre esses valores. Para cada combinação, o modelo é treinado e avaliado por validação cruzada, produzindo uma medida média de desempenho. Ao final, a combinação que apresenta o melhor resultado médio é selecionada.

No contexto deste trabalho, o `GridSearchCV` foi utilizado para realizar a escolha sistemática da melhor configuração de cada classificador dentro do **laço interno** da validação cruzada aninhada.

No laço interno, o `GridSearchCV` avaliou as combinações de hiperparâmetros utilizando múltiplas métricas (`accuracy`, `f1` e `roc_auc`), sendo a etapa de `refit` realizada com base no **F1-score**. Assim, a escolha final da melhor configuração em cada iteração priorizou o desempenho em F1 no conjunto de treino interno.

Foram utilizados dois níveis de validação cruzada:

- **Outer CV** (*laço externo*, para avaliação): $K_{outer} = 5$ folds estratificados;
- **Inner CV** (*laço interno*, para seleção de hiperparâmetros): $K_{inner} = 3$ folds estratificados.

O termo **stratified** indica que a proporção das classes é preservada em cada fold. Isso é especialmente importante em problemas de classificação, principalmente quando existe desbalanceamento entre as classes, como ocorre na base Adult Census Income.

Funcionamento da estratégia

1. O conjunto de dados é inicialmente dividido em 5 folds estratificados no **laço externo**;
2. Em cada iteração do laço externo, um fold é separado como **teste externo**, enquanto os outros 4 folds formam o **treino externo**;
3. Dentro desse treino externo, é executado um novo processo de validação cruzada com 3 folds estratificados, caracterizando o **laço interno**;
4. No laço interno, o `GridSearchCV` testa as diferentes combinações de hiperparâmetros previamente definidas para cada algoritmo;

5. Para cada combinação testada, o modelo é treinado e validado nos folds internos, produzindo um desempenho médio;
6. A combinação de hiperparâmetros que apresentar o melhor desempenho médio no laço interno é selecionada;
7. Em seguida, o modelo é treinado novamente utilizando todo o conjunto de **treino externo** e a melhor parametrização encontrada;
8. Por fim, esse modelo ajustado é avaliado no **fold externo**, que permaneceu isolado durante todo o processo de escolha dos hiperparâmetros;
9. O procedimento é repetido até que cada um dos 5 folds externos tenha sido usado uma vez como teste;
10. Ao final, as métricas obtidas nos folds externos são agregadas e apresentadas como **média ± desvio padrão**.

Intuição da abordagem

Essa estratégia pode ser entendida como a combinação de duas perguntas diferentes:

- o **laço interno** responde: *qual parametrização parece melhor?*
- o **laço externo** responde: *esse modelo, já ajustado, realmente generaliza bem para dados não vistos?*

Essa separação é importante porque evita uma estimativa excessivamente otimista de desempenho. Se os mesmos dados usados para ajustar hiperparâmetros também fossem usados para avaliar o modelo final, haveria risco de superestimar sua qualidade. Portanto, o **Nested Stratified K-Fold** fornece uma avaliação mais honesta e metodologicamente mais adequada.

3.5 Métricas de avaliação

Foram utilizadas três métricas de avaliação: **Accuracy**, **F1-score** e **ROC-AUC**. A escolha dessas métricas foi motivada pela necessidade de avaliar os modelos de forma mais completa, já que uma única métrica nem sempre é suficiente para descrever bem o desempenho de um classificador.

Nas fórmulas a seguir:

- TP (*True Positives*): instâncias positivas corretamente classificadas;
- TN (*True Negatives*): instâncias negativas corretamente classificadas;
- FP (*False Positives*): instâncias negativas classificadas incorretamente como positivas;
- FN (*False Negatives*): instâncias positivas classificadas incorretamente como negativas.

Acurácia (Accuracy)

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN}$$

A acurácia mede a proporção total de acertos do classificador. Ela foi utilizada por ser uma métrica simples, intuitiva e amplamente empregada em problemas de classificação. No entanto, sua interpretação isolada pode ser limitada em bases desbalanceadas, pois um modelo pode apresentar alta acurácia mesmo tendo desempenho fraco na classe minoritária. Por isso, a acurácia foi utilizada neste trabalho em conjunto com outras métricas.

Precisão, Revocação e F1-score

$$\begin{aligned}\text{Precision} &= \frac{TP}{TP + FP} & \text{Recall} &= \frac{TP}{TP + FN} \\ \text{F1} &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}\end{aligned}$$

A precisão indica, entre as instâncias previstas como positivas, quantas realmente são positivas. A revocação (*recall*) indica, entre as instâncias positivas reais, quantas foram corretamente identificadas pelo modelo. Já o F1-score combina precisão e revocação em uma única medida.

O **F1-score** foi incluído porque ele é mais informativo do que a acurácia quando há desbalanceamento entre as classes, como ocorre na base Adult Census Income. Como essa métrica considera simultaneamente erros do tipo falso positivo e falso negativo, ela permite avaliar melhor o desempenho do modelo na classe positiva.

ROC-AUC

A AUC (*Area Under the ROC Curve*) corresponde à área sob a curva ROC, que relaciona a taxa de verdadeiros positivos e a taxa de falsos positivos ao variar o limiar de decisão:

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN}$$

A ROC-AUC foi utilizada porque fornece uma medida da capacidade de separação entre as classes, independentemente de um único limiar fixo de classificação. Em termos intuitivos, quanto maior a AUC, melhor a capacidade do modelo de distinguir instâncias positivas e negativas.

Essa métrica é útil neste trabalho porque complementa a acurácia e o F1-score: enquanto a acurácia resume os acertos totais e o F1-score enfatiza o equilíbrio entre precisão e revocação, a ROC-AUC avalia a qualidade geral da discriminação entre as classes.

Motivação da escolha das métricas: As métricas **Accuracy**, **F1-score** e **ROC-AUC** foram escolhidas por serem complementares. A acurácia fornece uma visão geral do número de acertos, o F1-score é mais adequado para analisar desempenho em cenários com possível desbalanceamento, e a ROC-AUC mede a capacidade global de separação entre as classes. Dessa forma, a avaliação dos modelos não fica restrita a apenas um único critério.

3.6 Parametrizações testadas (por algoritmo)

Como o enunciado exige o teste de mais de uma configuração por algoritmo, foram definidos **grids de hiperparâmetros** para cada classificador. Em outras palavras, para cada modelo foi estabelecido previamente um conjunto de valores possíveis para alguns hiperparâmetros, permitindo que o **GridSearchCV** avaliasse sistematicamente diferentes combinações dentro do laço interno da validação cruzada aninhada.

A escolha desses grids buscou equilibrar dois objetivos: (i) explorar configurações relevantes para o comportamento de cada algoritmo e (ii) manter o custo computacional viável, especialmente no caso do Random Forest.

KNN

- `n_neighbors` $\in \{5, 15, 31\}$
- `weights` $\in \{\text{uniform}, \text{distance}\}$
- `p` $\in \{1, 2\}$ (distância de Minkowski, correspondendo às variantes Manhattan e Euclidiana)

No KNN, foram testados diferentes tamanhos de vizinhança, formas de ponderação e métricas de distância, pois essas escolhas influenciam diretamente a sensibilidade do modelo a padrões locais e ruídos nos dados.

Decision Tree

- `criterion` $\in \{\text{gini}, \text{entropy}\}$
- `max_depth` $\in \{\text{None}, 20\}$
- `min_samples_leaf` $\in \{1, 5\}$

Na Árvore de Decisão, foram avaliadas combinações relacionadas ao critério de divisão e ao controle de complexidade da árvore, uma vez que árvores muito profundas podem apresentar maior risco de overfitting. Além disso, foi utilizado `class_weight="balanced"` para reduzir o impacto de possíveis desbalanceamentos entre as classes.

Random Forest

- `n_estimators` $\in \{200, 500\}$
- `max_depth` $\in \{\text{None}, 20\}$
- `min_samples_leaf` $\in \{1, 5\}$

No Random Forest, foram testadas configurações relacionadas principalmente à quantidade de árvores, profundidade máxima e tamanho mínimo das folhas. Também foi utilizado `class_weight="balanced_subsample"`, buscando tornar o treinamento mais robusto diante de possíveis desbalanceamentos.

Observação: no contexto do **Nested Stratified K-Fold**, a seleção de hiperparâmetros foi realizada separadamente em cada iteração do laço externo, por meio do **GridSearchCV** no laço interno. Assim, a melhor combinação de hiperparâmetros pode variar entre os folds. Por esse motivo, o foco principal da análise recai sobre o **desempenho agregado dos modelos**, e não sobre a definição de uma única parametrização vencedora global.

Capítulo 4

Resultados e Discussão

4.1 Tabela-síntese dos resultados (média ± desvio padrão)

A Tabela 4.1 sumariza o desempenho dos modelos ao longo dos 5 folds externos do Nested CV. Em cada caso, a **média** representa o desempenho médio observado nos folds externos, enquanto o **desvio padrão** indica a variabilidade desse desempenho entre as partições avaliadas.

Tabela 4.1: Resultados (Nested Stratified CV: outer=5, inner=3). Valores em média ± desvio padrão. Os melhores resultados por métrica, dentro de cada base, estão destacados em negrito.

Dataset	Modelo	Accuracy	F1	ROC-AUC
Adult Census Income	KNN	0.8419 ± 0.0029	0.6480 ± 0.0059	0.8928 ± 0.0071
Adult Census Income	Decision Tree	0.8140 ± 0.0048	0.6728 ± 0.0046	0.8340 ± 0.0018
Adult Census Income	Random Forest	0.8347 ± 0.0048	0.7037 ± 0.0077	0.9138 ± 0.0034
Bank Marketing	KNN	0.8048 ± 0.0036	0.7833 ± 0.0037	0.8857 ± 0.0051
Bank Marketing	Decision Tree	0.7941 ± 0.0052	0.7856 ± 0.0054	0.8495 ± 0.0034
Bank Marketing	Random Forest	0.8429 ± 0.0068	0.8405 ± 0.0074	0.9126 ± 0.0026

De modo geral, os desvios padrão observados foram baixos a moderados, sugerindo desempenho relativamente estável entre os folds externos. Em particular, o Random Forest manteve desempenho consistente (baixa variabilidade) nas duas bases.

4.2 Discussão por base de dados

4.2.1 Adult Census Income

No dataset Adult Census Income, a classe positiva (>50K) é minoritária, o que torna a **acurácia isolada** uma métrica potencialmente enganosa. Nesse cenário, métricas como **F1-score** e **ROC-AUC** se tornam particularmente relevantes, pois ajudam a avaliar não apenas o número total de acertos, mas também a capacidade do modelo de identificar adequadamente a classe positiva e separar as classes de forma mais consistente.

Observou-se que:

- O **KNN** atingiu a **maior acurácia média** (0.8419), porém apresentou o **menor F1-score** (0.6480). Isso é compatível com o efeito do desbalanceamento: o modelo pode maximizar acertos na classe majoritária e ainda assim perder desempenho na identificação da classe positiva, reduzindo o equilíbrio entre precisão e revocação.
- A **Árvore de Decisão** apresentou F1-score intermediário (0.6728), mas **ROC-AUC** mais baixo (0.8340), sugerindo menor capacidade global de discriminação entre classes, possivelmente por maior sensibilidade a variações no conjunto de treino.
- O **Random Forest** apresentou o melhor desempenho em **F1-score** (0.7037) e **ROC-AUC** (0.9138), resultado consistente com a redução de variância promovida pelo ensemble e com melhor modelagem de interações entre atributos.

Assim, embora o KNN tenha obtido a maior acurácia média, o **Random Forest** apresentou o **melhor desempenho global** para a base Adult Census Income, principalmente quando se consideram métricas mais adequadas ao desbalanceamento.

Modelo de maior destaque na base Adult Census Income: Random Forest.

4.2.2 Bank Marketing

Na base Bank Marketing, as classes são quase balanceadas, o que torna a **acurácia** mais informativa do que no dataset Adult. Ainda assim, a análise conjunta com **F1-score** e **ROC-AUC** é importante para avaliar equilíbrio entre erros e capacidade de separação entre classes.

Observou-se que:

- **KNN** apresentou Accuracy (0.8048) e ROC-AUC (0.8857) superiores às da Árvore de Decisão, mas F1-score muito próximo (0.7833).
- A **Árvore de Decisão** obteve F1-score semelhante ao KNN (0.7856), porém com ROC-AUC menor (0.8495), indicando menor capacidade global de discriminação entre as classes.
- O **Random Forest** obteve os melhores resultados nas três métricas: **Accuracy** (0.8429), **F1-score** (0.8405) e **ROC-AUC** (0.9126). Isso reforça a vantagem do ensemble em bases tabulares heterogêneas, produzindo um modelo mais estável e com melhor generalização.

Modelo de maior destaque na base Bank Marketing: Random Forest.

4.3 Comparação geral: houve um algoritmo melhor nas duas bases?

Considerando os resultados obtidos nas duas bases, o **Random Forest** foi o modelo de maior destaque geral. Na base Adult Census Income, ele não apresentou a maior acurácia, mas obteve os melhores resultados em **F1-score** e **ROC-AUC**, que são particularmente relevantes quando há desbalanceamento. Já na base Bank Marketing, o Random Forest apresentou o melhor desempenho nas três métricas analisadas (**Accuracy**, **F1-score** e **ROC-AUC**).

Dessa forma, a análise conjunta indica que o **Random Forest** apresentou o **melhor desempenho global** entre os modelos avaliados, em linha com a fundamentação teórica de métodos *ensemble*, que tendem a reduzir variância e melhorar a capacidade de generalização.

Capítulo 5

Uso de LLMs

Conforme exigido no enunciado, declaramos que utilizamos uma LLM como auxílio em partes do trabalho (organização do pipeline, depuração de erros e estruturação textual do relatório). A LLM utilizada foi o: **ChatGPT 5.2 (OpenAI)**.

5.1 Onde utilizamos

- Ajuda na **depuração** de erros de implementação (ex.: inconsistências no rótulo positivo e mapeamento de classes);
- Sugestões para **estruturação de pipeline** (pré-processamento com ColumnTransformer + Pipeline);
- Apoio na **estruturação do relatório** para cobrir todos os itens solicitados.

5.2 Prompts utilizados (exemplos)

- “Recebi o erro `ValueError: invalid literal for int() with base 10: '<=50K'` no KNN dentro do GridSearch. Explique a causa e corrija o mapeamento do target para 0/1 sem quebrar o pipeline.”
- “Gere o esqueleto de um relatório em LaTeX contendo: introdução/objetivos; descrição das duas bases (instâncias, atributos, tipos, ausentes); metodologia (pré-processamento, ferramenta, algoritmos, divisão treino/teste, métricas com fórmulas, parâmetros); discussão e tabela-síntese.”
- “Para o terceiro classificador, o que exige não termos estudado em sala de aula, escolhemos o Random Forest. Me explique o seu funcionamento e me forneça um trecho de código em Python que mostre como ele opera.”

Observação: apesar do auxílio, nós revisamos criticamente o conteúdo e interpretamos os resultados.

Referências Bibliográficas

- [1] Pedregosa, F. et al. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 2011. (Documentação e implementações utilizadas via scikit-learn).
- [2] Breiman, L. *Random Forests*. Machine Learning, 45, 5–32, 2001.