



**Engenharia de Computação
Inteligência Artificial
09 - K-Nearest Neighbors (KNN)
Preparação de Dados, Distância e
Classificação**

Organização da aula

- Retomada: aprendizagem supervisionada e Árvore de Decisão.
- Dados reais: inspeção, tipos de atributos e qualidade dos dados.
- Valores ausentes, atributos categóricos, atributos numéricos e outliers.
- Preparação de um CSV real e treinamento de uma árvore já conhecida.
- K-Nearest Neighbors (KNN): ideia, distância, vizinhança e votação.
- Influência da escala dos atributos e padronização.
- Escolha de k por validação cruzada k -fold e curva de desempenho.
- Pseudoalgoritmo e implementação com *Scikit-learn*.

Fio condutor

O que precisa acontecer entre receber dados brutos e utilizar a proximidade entre exemplos para classificar um novo caso?

Retomada: o que fizemos na Aula 08?

Retomando

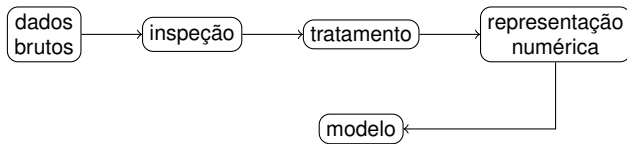
Na Aula 08, partimos de exemplos rotulados e induzimos uma Árvore de Decisão capaz de classificar novos exemplos.

- separamos atributos X e alvo y ;
- dividimos os exemplos em treinamento e teste;
- usamos Gini ou Entropia para escolher divisões;
- `fit()` construiu a árvore;
- `predict()` utilizou a árvore aprendida;
- `max_depth` permitiu controlar a complexidade do modelo.

Pergunta

O que acontece quando os dados não chegam prontos como no conjunto Iris?

Antes do algoritmo: os dados



Mensagem central

Em problemas reais, o algoritmo de aprendizagem é apenas uma etapa. Antes dele, precisamos entender o que cada coluna significa e como os dados devem ser representados.

Obtendo a base Titanic

- Nesta aula utilizaremos a base **Titanic - Machine Learning from Disaster**, disponibilizada pelo Kaggle.
- Na aba **Data**, a competição disponibiliza três arquivos principais:
 - `train.csv` - exemplos com a coluna-alvo `Survived`;
 - `test.csv` - passageiros sem o rótulo de sobrevivência;
 - `gender_submission.csv` - exemplo de arquivo de submissão.
- Para a prática desta aula utilizaremos **somente o arquivo `train.csv`**.

Onde baixar?

kaggle.com/competitions/titanic/data

Acesso ao Kaggle

O download pode exigir login no Kaggle e aceite das regras da competição. Faça o download do `train.csv` antes de iniciar a prática.

Estudo de caso: Titanic

Utilizaremos o arquivo `train.csv` da base Titanic como exemplo de dados tabulares reais.

Coluna	Tipo inicial	Questão de preparação
Survived	binária	alvo da classificação
Pclass	discreta/ordinal	manter como atributo
Sex	categórica	transformar em representação numérica
Age	numérica	possui valores ausentes
Fare	numérica	possui valores extremos
Embarked	categórica	possui valores ausentes
Name/Ticket/Cabin	texto/categoria	avaliar utilidade e tratamento

Pergunta

Todas as colunas devem entrar no modelo apenas porque estão disponíveis no CSV?

Primeira inspeção do CSV

```
import pandas as pd

df = pd.read_csv("train.csv")

print(df.head())
print(df.info())
print(df.isna().sum())
print(df.describe())
```

Tipos de atributos encontrados

Numéricos

Representam quantidades ou valores sobre os quais operações numéricas fazem sentido.

- Age
- Fare
- SibSp
- Parch

Catégoricos/textuais

Representam categorias, identificadores ou descrições.

- Sex
- Embarked
- Name
- Ticket

Observação

O tipo armazenado no arquivo não determina sozinho o papel semântico do atributo. `Pclass`, por exemplo, é numérico no CSV, mas representa uma categoria ordenada.

Campos ausentes: não existe uma única resposta

Quando um valor está ausente, algumas estratégias possíveis são:

- remover a linha;
- remover a coluna;
- preencher usando média, mediana ou moda;
- criar uma categoria específica para “desconhecido”;
- utilizar um método de imputação mais elaborado.

Decisão depende do contexto

Devemos tratar Age, Cabin e Embarked da mesma maneira apenas porque possuem dados ausentes?

Outlier não significa erro

Considere F_{are} , o valor da tarifa paga.

- a maior parte dos passageiros pode estar concentrada em uma faixa menor;
- alguns valores podem estar muito distantes do restante;
- um valor extremo pode ser legítimo, e não um erro de digitação;
- remover automaticamente pontos extremos pode apagar informação importante.

Antes de remover

O valor é impossível, inconsistente ou apenas raro?

Inspecionando valores extremos

```
print(df["Fare"].describe())

import matplotlib.pyplot as plt

plt.boxplot(df["Fare"].dropna())
plt.ylabel("Fare")
plt.show()
```

Observação

O boxplot é uma ferramenta de inspeção. Ele ajuda a identificar observações distantes, mas não decide automaticamente se elas devem ser removidas.

Colunas que talvez não devam entrar diretamente

- `PassengerId`: identificador único; pode não representar uma característica generalizável;
- `Name`: texto livre com informação potencial, mas exige engenharia de atributos;
- `Ticket`: alta cardinalidade e estrutura textual;
- `Cabin`: muitos valores ausentes e várias categorias;
- `Survived`: é o alvo e não pode aparecer entre os atributos de entrada.

Vazamento de informação

O que aconteceria se uma coluna que revelasse diretamente a resposta fosse mantida entre os atributos de entrada?

Palavras precisam virar números

Algoritmos do Scikit-learn trabalham com uma representação numérica dos atributos.

Codificação inadequada

Embarked:

C $\rightarrow$ 0

Q $\rightarrow$ 1

S $\rightarrow$ 2

Isso introduz uma ordem artificial.

One-hot encoding

Cria indicadores separados:

Embarked_C

Embarked_Q

Embarked_S

Observação

Para uma variável binária, uma codificação 0/1 pode ser suficiente. Para categorias nominais com mais valores, one-hot encoding evita impor uma ordem inexistente.

Uma decisão importante: separar antes de aprender transformações

dados $\rightarrow$ treino/teste $\rightarrow$ aprender preparação
no treino $\rightarrow$ aplicar no teste

Evitar vazamento de dados

Estatísticas usadas na preparação, como mediana ou média, devem ser estimadas apenas a partir do conjunto de treinamento. O conjunto de teste deve representar dados ainda não vistos.

Pergunta

Se calcularmos a mediana usando também o conjunto de teste, o modelo continua completamente isolado desses dados?

Selecionando atributos e separando treino/teste

```
from sklearn.model_selection import train_test_split

features = [
    "Pclass", "Sex", "Age", "SibSp",
    "Parch", "Fare", "Embarked"
]

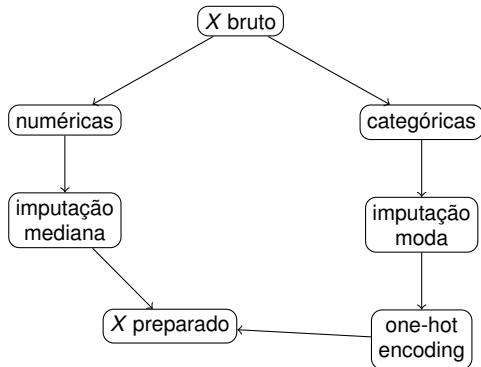
X = df[features]
y = df["Survived"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42
)
```

Mesma estrutura da Aula 08

A diferença é que agora x ainda contém valores ausentes e atributos categóricos. Precisamos preparar os dados antes de treinar o modelo.

Preparação por tipo de coluna



Observação

Colunas diferentes podem exigir transformações diferentes. O `ColumnTransformer` permite organizar esse fluxo em um único objeto.

Pipeline de preparação

```
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder

numeric_features = ["Pclass", "Age", "SibSp", "Parch", "Fare"]
categorical_features = ["Sex", "Embarked"]

numeric_pipe = Pipeline([
    ("imputer", SimpleImputer(strategy="median"))
])

categorical_pipe = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])
```

O que acontece com os atributos categóricos?

Antes da transformação

Sex	Embarked
male	S
female	C
female	S
male	Q

Depois do one-hot encoding

Sex_F	Sex_M	Emb_C	Emb_Q	Emb_S
0	1	0	0	1
1	0	1	0	0
1	0	0	0	1
0	1	0	1	0

No nosso Pipeline

`Sex` e `Embarked` passam primeiro pela imputação dos valores ausentes e depois pelo `OneHotEncoder`.

Observação

O `OneHotEncoder` evita criar uma ordem artificial entre categorias. Por exemplo, não faz sentido interpretar `Cherbourg = 0`, `Queenstown = 1` e `Southampton = 2` como uma relação de ordem.

Como o código faz esse tratamento?

```
categorical_features = ["Sex", "Embarked"]

categorical_pipe = Pipeline([
    ("imputer",
     SimpleImputer(strategy="most_frequent")),

    ("onehot",
     OneHotEncoder(handle_unknown="ignore"))
])
```

- SimpleImputer: substitui valores ausentes pela categoria mais frequente;
- OneHotEncoder: transforma cada categoria em uma coluna binária;
- handle_unknown="ignore": evita erro caso apareça uma categoria nova no conjunto de teste.

Interpretação

O modelo recebe diretamente os textos male, female, C, Q e S?

Combinando preparação e Árvore de Decisão

```
from sklearn.tree import DecisionTreeClassifier

preprocess = ColumnTransformer([
    ("num", numeric_pipe, numeric_features),
    ("cat", categorical_pipe, categorical_features)
])

model = Pipeline([
    ("preprocess", preprocess),
    ("classifier", DecisionTreeClassifier(
        max_depth=4,
        random_state=42
    ))
])

model.fit(X_train, y_train)
```

Por que usar Pipeline?

A preparação é ajustada no conjunto de treinamento e aplicada de forma consistente ao conjunto de teste e a futuros exemplos.

Predição e avaliação da árvore

```
from sklearn.metrics import accuracy_score

y_pred = model.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print(f"Acuracia: {acc:.3f}")
```

Retomando

O algoritmo de classificação continua sendo o mesmo da Aula 08. O novo elemento foi construir corretamente a representação de entrada a partir de dados brutos.

Pergunta

O que mudou: a Árvore de Decisão ou a forma como preparamos os dados para ela?

Atividade curta - decisões de preparação

Em duplas - 7 minutos

Para cada coluna, proponha uma decisão inicial e justifique-a.

Coluna	Decisão	Justificativa
Age		
Cabin		
Sex		
Embarked		
Fare		
PassengerId		

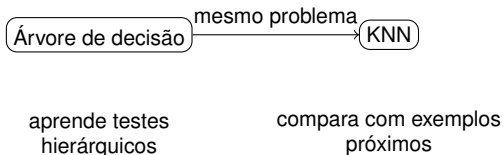
Atividade curta - uma solução possível

- Age: imputar mediana;
- Cabin: remover na primeira versão do experimento;
- Sex: codificar como variável binária ou one-hot;
- Embarked: imputar moda + one-hot encoding;
- Fare: manter inicialmente e investigar valores extremos;
- PassengerId: remover por ser identificador.

Não existe uma única solução

Outras decisões podem ser corretas, desde que tenham justificativa e não introduzam vazamento de informação.

Da árvore para outro modo de classificar



Nova estratégia

E se, em vez de aprender regras de decisão, classificássemos um novo exemplo observando os exemplos de treinamento mais próximos dele?

O que é K-Nearest Neighbors?

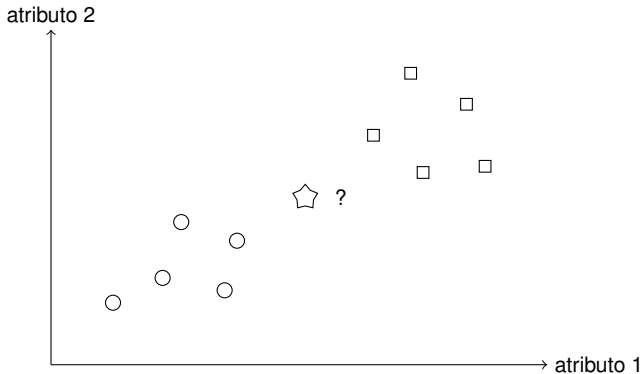
K-Nearest Neighbors (KNN) é um método de aprendizagem supervisionada baseado em proximidade entre exemplos.

- recebe um exemplo ainda não rotulado;
- mede sua distância aos exemplos de treinamento;
- seleciona os k vizinhos mais próximos;
- utiliza os rótulos desses vizinhos para decidir a classe.

Característica importante

O KNN é frequentemente chamado de método baseado em instâncias ou *lazy learning*: grande parte do trabalho ocorre no momento da predição, e não na construção de um modelo explícito como uma árvore.

Exemplo geométrico



Pergunta

Sem construir uma árvore, que informação do conjunto de treinamento parece útil para decidir a classe do novo ponto?

KNN em cinco passos

Para classificar um novo exemplo x :

- 1 calcular a distância entre x e cada exemplo de treinamento;
- 2 ordenar os exemplos pela distância;
- 3 selecionar os k vizinhos mais próximos;
- 4 observar os rótulos desses vizinhos;
- 5 atribuir a classe mais votada.

Parâmetro do método

O que muda quando usamos $k = 1$, $k = 3$ ou $k = 15$?

Pseudoalgoritmo do KNN

Input: exemplo x , conjunto de treinamento D , número de vizinhos k

Output: classe prevista para x

criar lista vazia de distâncias

foreach $(x_i, y_i) \in D$ **do**

 calcular $d(x, x_i)$

 armazenar $(d(x, x_i), y_i)$

end foreach

ordenar a lista pela distância

selecionar os k primeiros elementos

contar os rótulos presentes nesses vizinhos

return classe com maior número de votos

Distância Euclidiana

Para dois exemplos com n atributos:

$$d(x, z) = \sqrt{\sum_{j=1}^n (x_j - z_j)^2}.$$

Para dois atributos:

$$d(x, z) = \sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2}.$$

Observação

A distância Euclidiana é uma escolha comum, mas não é a única. A métrica deve ser coerente com a representação dos dados e com o problema.

Exemplo manual de distância

Considere o novo exemplo:

$$x = (4, 3)$$

e dois exemplos de treinamento:

$$a = (2, 2), \quad b = (7, 5).$$

Então:

$$d(x, a) = \sqrt{(4 - 2)^2 + (3 - 2)^2} = \sqrt{5} \approx 2,24$$

$$d(x, b) = \sqrt{(4 - 7)^2 + (3 - 5)^2} = \sqrt{13} \approx 3,61$$

Pergunta

Qual dos dois exemplos exerce maior influência se $k = 1$?

Atividade - classifique manualmente

Individual - 5 minutos

Um novo exemplo $x = (4, 3)$ deve ser classificado usando $k = 3$.

Exemplo	Coordenadas	Classe	Distância a x
A	(2, 2)	círculo	?
B	(3, 4)	círculo	?
C	(5, 3)	quadrado	?
D	(7, 5)	círculo	?
E	(6, 1)	quadrado	?

Pergunta

Quais são os três vizinhos mais próximos e qual classe será prevista?

Atividade - resposta comentada

Para $x = (4, 3)$:

- $d(x, A) = \sqrt{5} \approx 2,24$;
- $d(x, B) = \sqrt{2} \approx 1,41$;
- $d(x, C) = 1$;
- $d(x, D) = \sqrt{13} \approx 3,61$;
- $d(x, E) = \sqrt{8} \approx 2,83$.

Os três vizinhos mais próximos são C, B e A.

Votação

C é quadrado; B e A são círculos. Portanto, para $k = 3$, a classe prevista é **círculo**.

O mesmo exemplo com diferentes valores de k

A ordem dos vizinhos, do mais próximo para o mais distante, é:

$$C_{\square} \rightarrow B_{\circ} \rightarrow A_{\circ} \rightarrow E_{\square} \rightarrow D_{\circ}$$

$k = 1$

Vizinho considerado:

$C_{\square}$

previsão: **quadrado**

$k = 3$

Vizinhos considerados:

$C_{\square}, B_{\circ}, A_{\circ}$

previsão: **círculo**

$k = 5$

Vizinhos considerados:

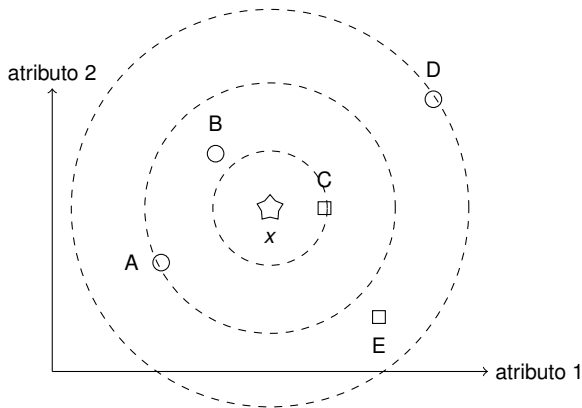
$C_{\square}, B_{\circ}, A_{\circ}, E_{\square}, D_{\circ}$

previsão: **círculo**

O que aprendemos?

Ao aumentar k , ampliamos a vizinhança considerada. A decisão pode mudar, mas não precisa alternar a cada novo valor de k .

Como k altera a vizinhança



Observação

Os círculos representam, de forma esquemática, vizinhanças progressivamente maiores. Quanto maior k , menos local tende a ser a decisão.

Por que a escala importa?

Considere dois atributos:

- idade: aproximadamente de 18 a 80;
- renda: aproximadamente de 1.000 a 50.000.

Na distância Euclidiana, diferenças em renda podem dominar completamente as diferenças em idade.

distância $\neq$ importância semântica

Pergunta

Se um atributo utiliza números muito maiores, ele deveria automaticamente ter mais influência na classificação?

Padronização dos atributos

Uma transformação comum é o *z-score*:

$$z = \frac{x - \mu}{\sigma}.$$

em que:

- μ é a média do atributo no conjunto de treinamento;
- σ é o desvio-padrão do atributo no conjunto de treinamento.

Observação

A padronização não torna os atributos “mais corretos”. Ela coloca suas escalas em condições mais comparáveis para métodos baseados em distância.

KNN com Pipeline no Scikit-learn

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

X, y = load_iris(return_X_y=True)

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42
)

model = make_pipeline(
    StandardScaler(),
    KNeighborsClassifier(n_neighbors=5)
)
```

Treinar e usar o KNN

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

- `StandardScaler.fit`: aprende média e desvio-padrão apenas no treino;
- `StandardScaler.transform`: aplica a transformação;
- `KNeighborsClassifier.fit`: armazena as informações necessárias dos exemplos de treino;
- `predict`: procura vizinhos para cada novo exemplo.

Observação

O significado de `fit()` depende do algoritmo. No KNN não surge uma árvore ou uma fronteira paramétrica explícita.

Avaliação inicial

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Acuracia: {accuracy:.3f}")
```

Observação

O conjunto de teste deve ser utilizado para estimar o desempenho final do modelo, e não para escolher continuamente os hiperparâmetros.

Pergunta

Então como escolher k sem ficar testando vários valores diretamente no conjunto de teste?

O papel de k

k pequeno

- decisão muito local;
- maior sensibilidade a ruído;
- fronteira mais irregular;
- risco de sobreajuste.

k grande

- decisão mais suavizada;
- vizinhança inclui pontos mais distantes;
- pode apagar estrutura local;
- risco de subajuste.

Compromisso

Como escolher uma região de valores de k que generalize melhor?

KFold versus StratifiedKFold

Em classificação, é importante observar também a distribuição das classes.

KFold

Divide os exemplos em k partes. Não garante que cada parte mantenha aproximadamente a mesma proporção das classes.

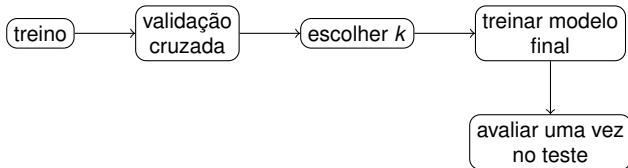
StratifiedKFold

Procura preservar em cada *fold* aproximadamente a proporção das classes observada no conjunto completo.

Observação

Para problemas de classificação, especialmente quando as classes são desbalanceadas, `StratifiedKFold` costuma ser uma escolha mais apropriada.

Validação cruzada para escolher k



Separação de papéis

O treinamento ajuda a ajustar o modelo; a validação ajuda a escolher hiperparâmetros; o teste estima o desempenho final em dados não utilizados nessas escolhas.

Curva de desempenho em função de k

```
from sklearn.model_selection import StratifiedKFold,
    cross_val_score

k_values = range(1, 31)
scores = []
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

for k in k_values:
    model = make_pipeline(
        StandardScaler(),
        KNeighborsClassifier(n_neighbors=k)
    )
    score = cross_val_score(
        model, X_train, y_train,
        cv=cv,
        scoring="accuracy"
    ).mean()
    scores.append(score)
```

Visualizando a curva de k

```
import matplotlib.pyplot as plt

plt.plot(list(k_values), scores, marker="o")
plt.xlabel("k")
plt.ylabel("Acuracia media de validacao")
plt.grid(True)
plt.show()
```

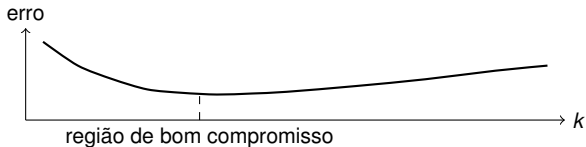
Sobre o “cotovelo”

A curva ajuda a identificar uma região estável de bom desempenho. Não existe uma regra universal exigindo escolher visualmente um único ponto de cotovelo.

Erro versus k

Também podemos visualizar:

$$\text{erro}(k) = 1 - \text{acurcia}(k).$$



Pergunta

Por que valores consecutivos de k podem produzir desempenhos muito semelhantes?

Escolher k não é decorar uma regra

Algumas heurísticas são úteis, mas não substituem validação.

- k ímpar pode reduzir a chance de empate em classificação binária, mas não elimina todos os empates possíveis;
- o melhor k depende da distribuição dos dados;
- número de exemplos, ruído, dimensionalidade e escala influenciam a vizinhança;
- escolher k olhando repetidamente o teste produz uma estimativa otimista do desempenho.

Observação

Hiperparâmetros devem ser escolhidos usando apenas informações permitidas durante o treinamento/validação.

Outras métricas de distância

Além da Euclidiana, existem outras formas de medir proximidade.

- Manhattan;
- Minkowski;
- Hamming para certos tipos de representação discreta;
- métricas específicas do domínio.

Nesta aula

Utilizaremos principalmente a distância Euclidiana. O objetivo é compreender como a escolha da representação e da escala influencia o conceito de “vizinho”.

Árvore de Decisão versus KNN

Aspecto	Árvore	KNN
ideia central	testes hierárquicos	proximidade entre exemplos
treinamento	constrói a árvore	armazena estrutura dos dados
predição escala	percorre nós geralmente pouco sensível	calcula/procura vizinhos fortemente relevante
interpretação	regras explícitas	decisão local por vizinhos
hiperparâmetro inicial	profundidade	número de vizinhos k

Erros conceituais frequentes

- remover automaticamente todo outlier;
- transformar categorias nominais em números que criam uma ordem artificial;
- calcular estatísticas de preparação usando treino e teste juntos;
- aplicar KNN em atributos com escalas muito diferentes sem analisar o efeito;
- escolher k pelo melhor resultado no conjunto de teste;
- acreditar que k ímpar garante ausência de empates em qualquer problema;
- interpretar `fit()` do KNN como a construção de uma árvore ou função paramétrica explícita.

Síntese do núcleo da aula

- dados reais precisam ser inspecionados antes do treinamento;
- valores ausentes, categorias e outliers exigem decisões justificadas;
- pipelines ajudam a evitar inconsistências e vazamento de dados;
- KNN classifica por proximidade entre exemplos;
- a métrica e a escala dos atributos determinam quem é considerado vizinho;
- k controla o grau de localidade da decisão;
- validação cruzada permite escolher k sem utilizar o conjunto de teste nessa decisão.

Cheque sua compreensão

Por que o mesmo conjunto de dados pode exigir preparação diferente para Árvore de Decisão e KNN?

Até aqui: núcleo da Aula 09

CSV $\rightarrow$ inspeção $\rightarrow$ preparação $\rightarrow$ distância $\rightarrow$
vizinhos $\rightarrow$ escolha de k $\rightarrow$ predição

Ponto de parada para o encontro de 100 minutos

Ao final do encontro, o estudante deve conseguir justificar decisões básicas de preparação de dados e explicar, executar e parametrizar um classificador KNN.

Antes de avançar

Você consegue explicar por que a padronização pode alterar completamente os vizinhos escolhidos pelo KNN?

Conexão com o Trabalho 2

A preparação desta aula será reutilizada no Trabalho 2.

- identificar quais colunas podem ser usadas como atributos;
- justificar o tratamento de ausentes e categóricos;
- evitar vazamento entre treino, validação e teste;
- selecionar hiperparâmetros sem usar o teste como ferramenta de ajuste;
- comparar o comportamento de algoritmos diferentes no mesmo problema.

Mais do que executar `fit()`

Uma boa solução precisa justificar a preparação dos dados e as decisões experimentais, não apenas apresentar a acurácia final.

Referências

- Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*. Seções sobre aprendizagem a partir de exemplos.
- Mitchell, T. *Machine Learning*. McGraw-Hill.
- Duda, R.; Hart, P.; Stork, D. *Pattern Classification*. Wiley.
- Scikit-learn. Documentação de `KNeighborsClassifier`, `StandardScaler`, `Pipeline`, `ColumnTransformer` e validação cruzada.
- Kaggle Titanic - exemplo de conjunto tabular para preparação de dados e classificação.

Dúvidas



Dúvida é o começo da sabedoria.