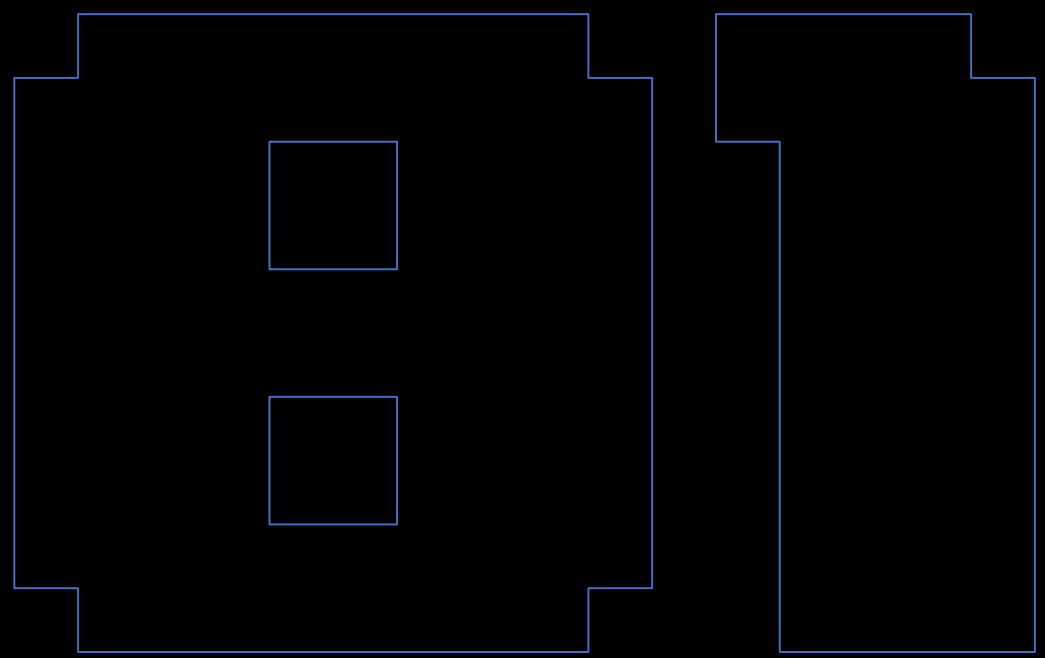
O FEITIÇO DO PYTHON

APRENDA MACHINE LEARNING

Gustavo Cigerza Padilha

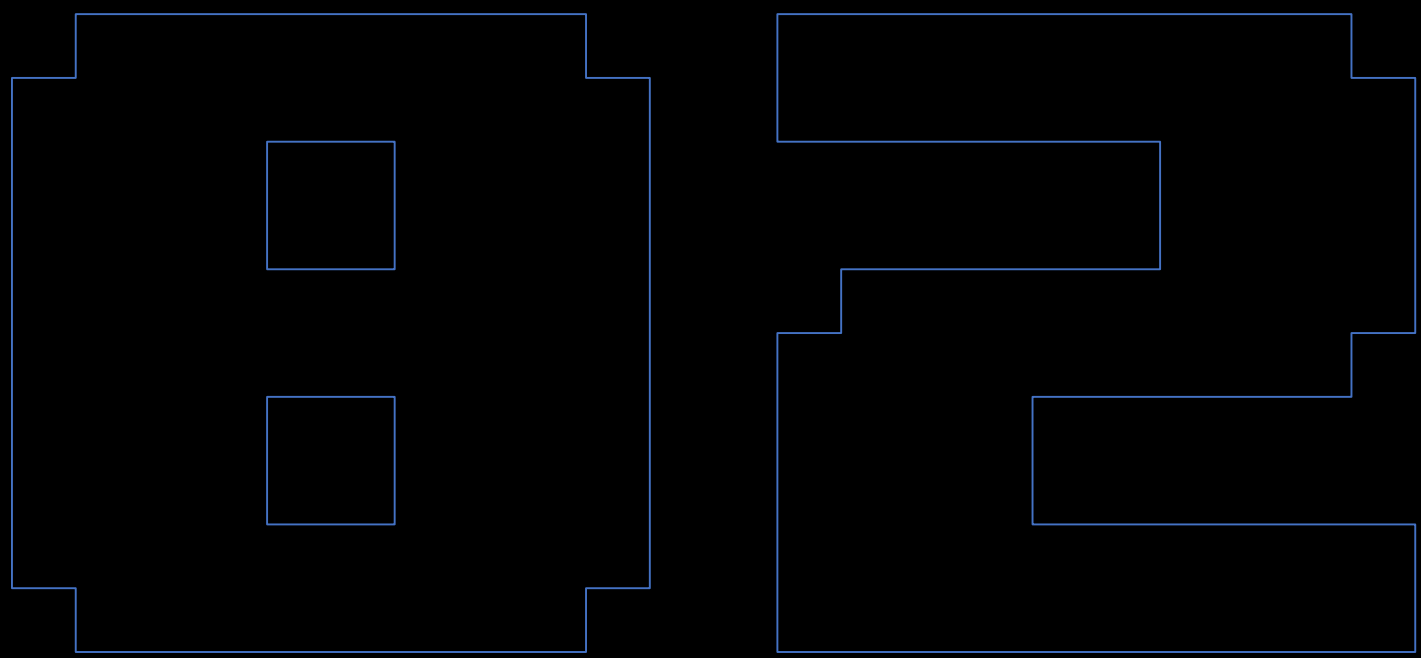


INTRODUÇÃO

🧠 MACHINE LEARNING COM PYTHON

Guia Simples e Prático

Machine Learning (Aprendizado de Máquina) é uma área da inteligência artificial que ensina o computador a aprender com dados — sem ser programado diretamente para cada tarefa. Python é a linguagem mais usada nessa área, por ser fácil de entender e ter ótimas bibliotecas.Neste eBook, você vai ver os principais conceitos de ML com Python de forma direta, com exemplos reais e claros. Ideal para quem quer começar a aplicar na prática.



COLETANDO E ENTENDENDO OS DADOS

📊 COLETANDO E ENTENDENDO OS DADOS

Todo projeto de Machine Learning começa com dados. Eles são a base para treinar, testar e avaliar modelos. Se os dados forem ruins, o resultado também será. Vamos carregar um dataset real de diabetes:

O FEITIÇO PYTHON - GUSTAVO.PY

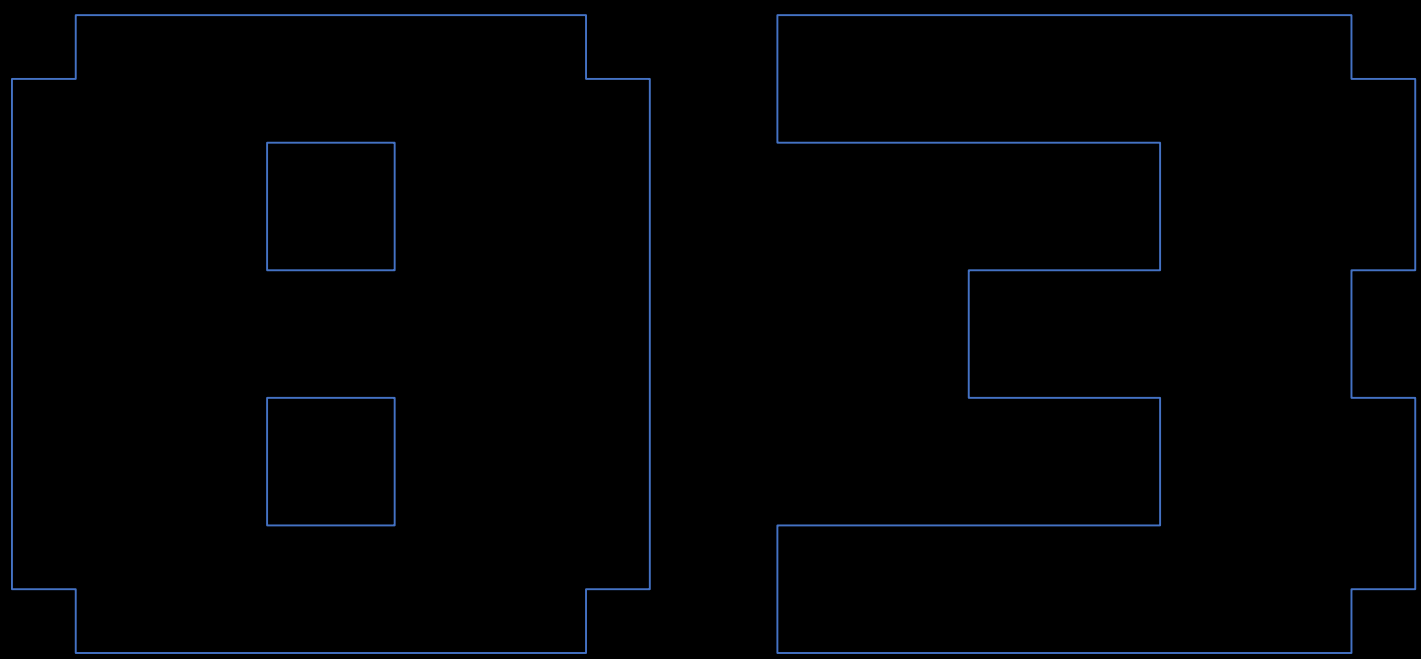
```python
import pandas as pd

# Carrega o dataset
df = pd.read_csv("diabetes.csv")

# Mostra as primeiras linhas
print(df.head())
```

📊 COLETANDO E ENTENDENDO OS DADOS

Aqui, cada linha representa um paciente, e cada coluna uma característica (idade, glicose, pressão, etc.). O objetivo é prever a coluna "Outcome", que indica se o paciente tem diabetes ou não.Antes de seguir, é importante conhecer os dados:

O FEITIÇO PYTHON - GUSTAVO.PY

```
print(df.info())        # Tipos de dados
print(df.describe())    # Estatísticas básicas
```



PREPARANDO OS DADOS

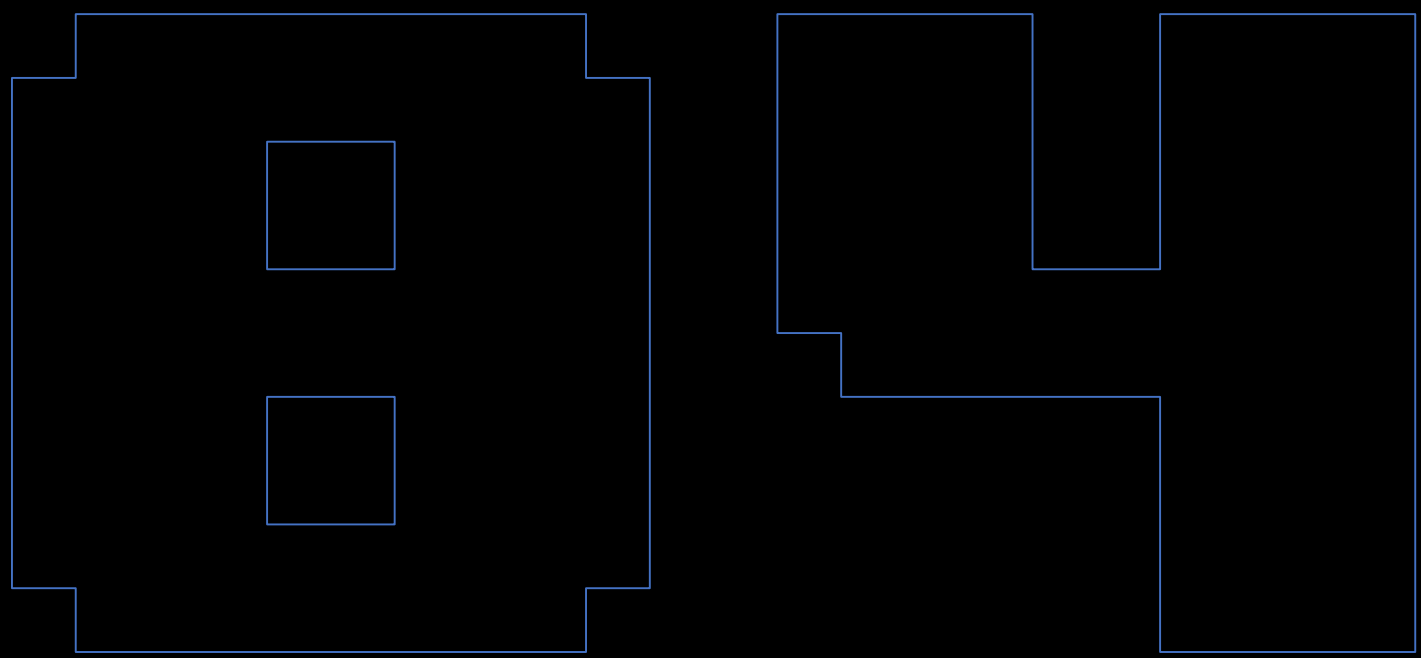
🧹 PREPARANDO OS DADOS

Limpeza e divisão

Não basta ter dados. Eles precisam estar prontos para o modelo entender. Isso inclui tratar valores nulos, normalizar e dividir os dados em treino e teste.
Essa divisão é essencial para testar o modelo com dados que ele nunca viu — como se fosse o mundo real.

O FEITIÇO PYTHON - GUSTAVO.PY

```python
from sklearn.model_selection
import train_test_split

X = df.drop("Outcome", axis=1)
# Atributos (features)

y = df["Outcome"]
# Alvo (target)

# Divide os dados
(80% treino, 20% teste)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2, random_state=42)
```



CRIANDO O PRIMEIRO MODELO

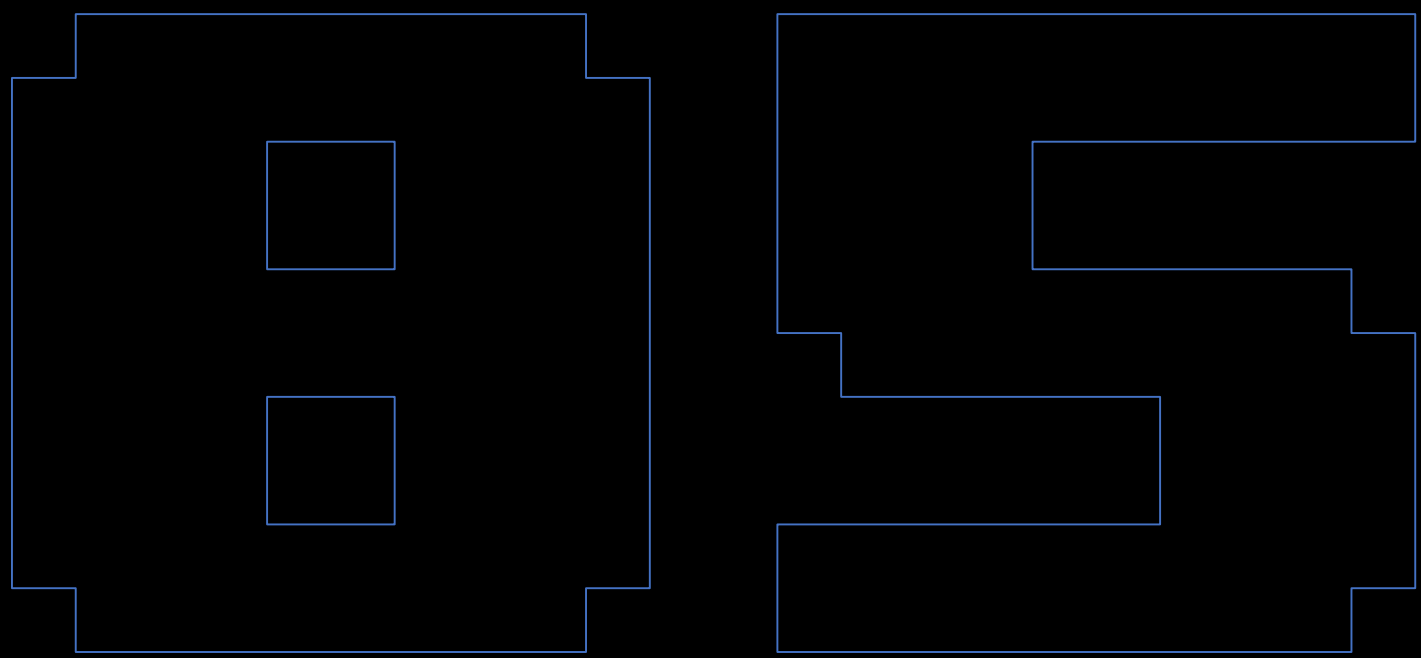
🧠 CRIANDO O PRIMEIRO MODELO

Classificação

Agora que os dados estão prontos, é hora de treinar um modelo. Vamos começar com um algoritmo simples e fácil de entender: KNN (K-Nearest Neighbors). Ele classifica um novo dado com base nos vizinhos mais próximos.

O FEITIÇO PYTHON - GUSTAVO.PY

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Acurácia:", accuracy_score(y_test, y_pred))
```



REGRESSÃO: PREVENDO VALORES NUMÉRICOS

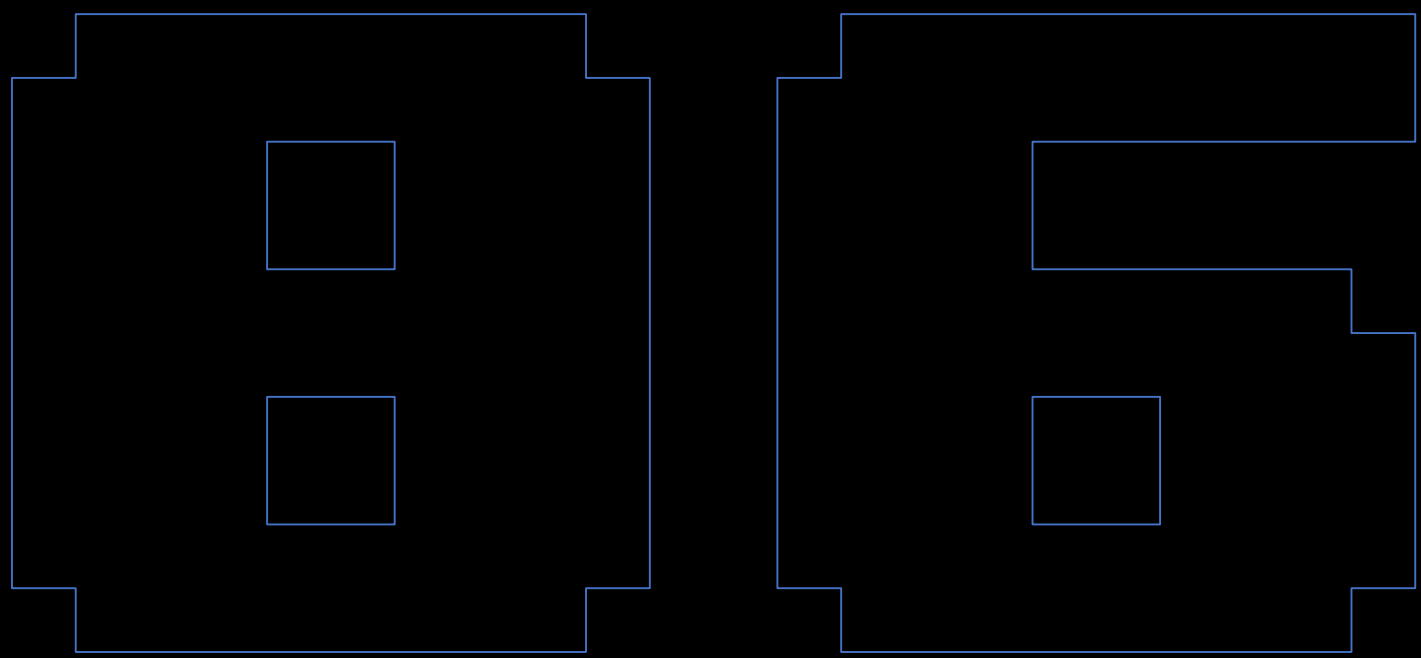
📈 PREVENDO VALORES NUMÉRICOS

Regressão

Nem todo problema é de "sim ou não". Às vezes, queremos prever números, como o preço de uma casa ou a glicose de um paciente. Isso é regressão.
Exemplo: prever a progressão de diabetes ao longo do tempo.
O R^2 Score mostra o quanto o modelo consegue explicar da variação nos dados (ideal é estar perto de 1).

O FEITIÇO PYTHON - GUSTAVO.PY

```python
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_diabetes

data = load_diabetes()
X, y = data.data, data.target

X_train, X_test, y_train, y_test = train_test_split(X, y)

reg = LinearRegression()
reg.fit(X_train, y_train)

print("R² Score:", reg.score(X_test, y_test))
```



AVALIANDO O MODELO (MÉTRICAS)

🧪 AVALIANDO O MODELO

Métricas

Treinar um modelo é fácil. Difícil é saber se ele está bom. Por isso usamos métricas.

Para classificação:

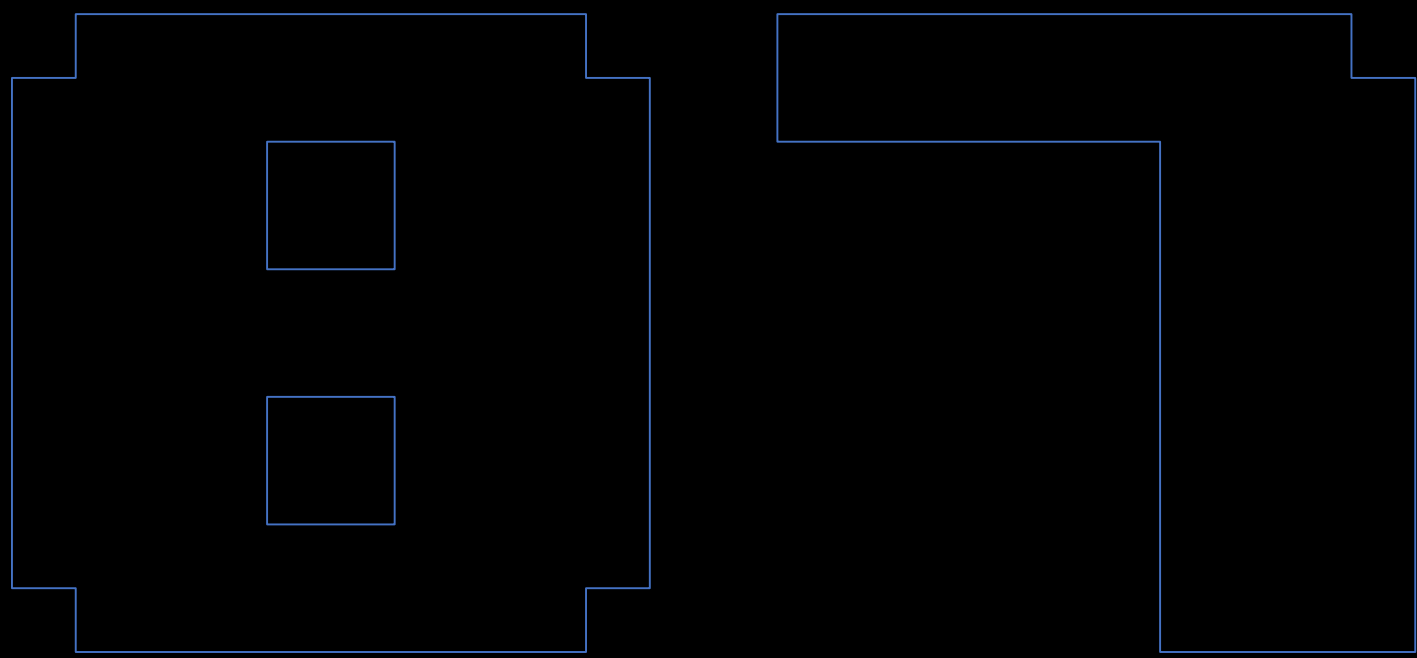
- Acurácia
- Matriz de confusão
- F1-score

Para regressão:

- R^2 (coeficiente de determinação)
- MAE (erro absoluto médio)
- MSE (erro quadrático médio)

O FEITIÇO PYTHON - GUSTAVO.PY

```python
from sklearn.metrics import confusion_matrix

print(confusion_matrix(y_test, y_pred))
```



AJUSTANDO O MODELO

⚙ 6. AJUSTANDO O MODELO

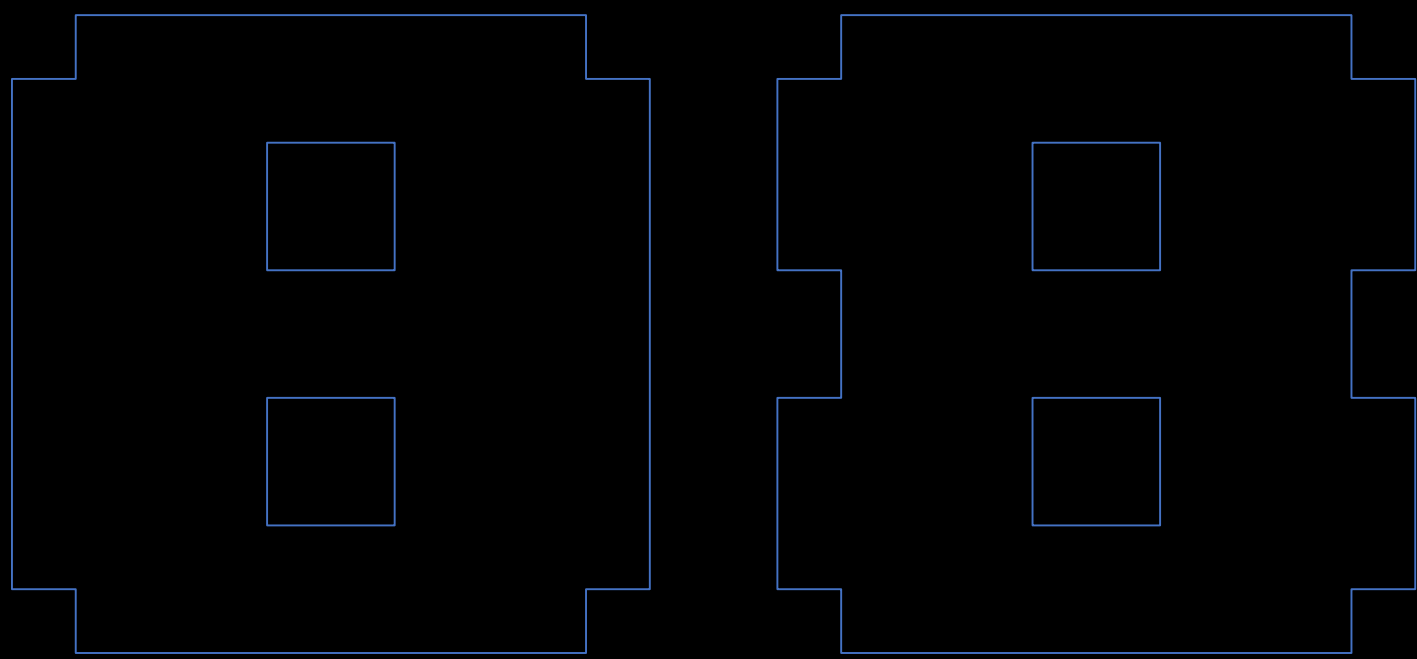
Hyperparameter Tuning

Modelos têm "configurações internas", chamadas hiperparâmetros. Mudar esses valores pode melhorar (ou piorar) o desempenho. Ao invés de adivinhar, usamos o GridSearchCV para testar várias opções automaticamente.

Esse processo é como testar várias receitas e escolher a mais saborosa.

O FEITIÇO PYTHON - GUSTAVO.PY

```python
from sklearn.model_selection import GridSearchCV

param_grid = {"n_neighbors": [3, 5, 7, 9]}
grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
grid.fit(X_train, y_train)

print("Melhor modelo:", grid.best_params_)
```



ESCOLHENDO O QUE IMPORTA

🔍 ESCOLHENDO O QUE IMPORTA

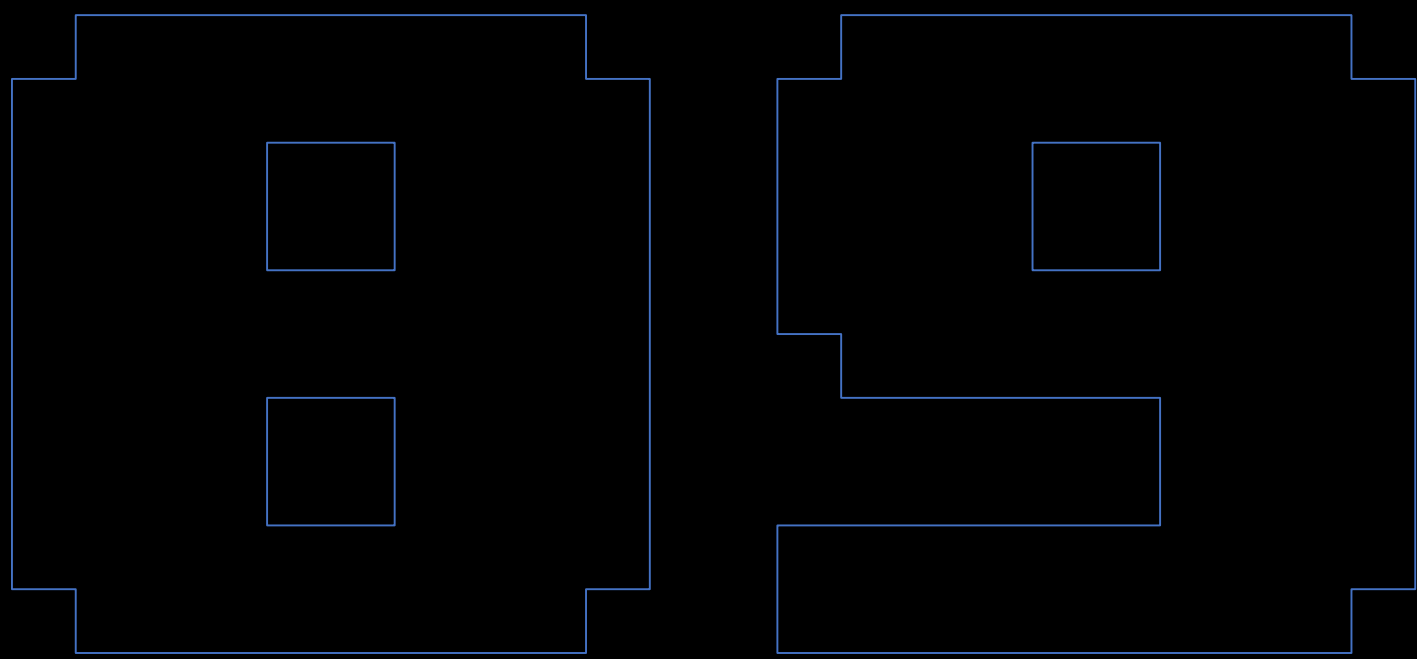
Seleção de Features

Nem toda coluna ajuda. Algumas podem confundir o modelo. A seleção de features ajuda a manter só o que realmente importa.

Isso pode acelerar o modelo e até melhorar a precisão.

O FEITIÇO PYTHON - GUSTAVO.PY

```python
from sklearn.feature_selection import SelectKBest, f_classif

selector = SelectKBest(score_func=f_classif, k=5)
X_new = selector.fit_transform(X, y)
```



MODELOS POPULARES PARA COMEÇAR

MODELOS POPULARES PARA COMEÇAR

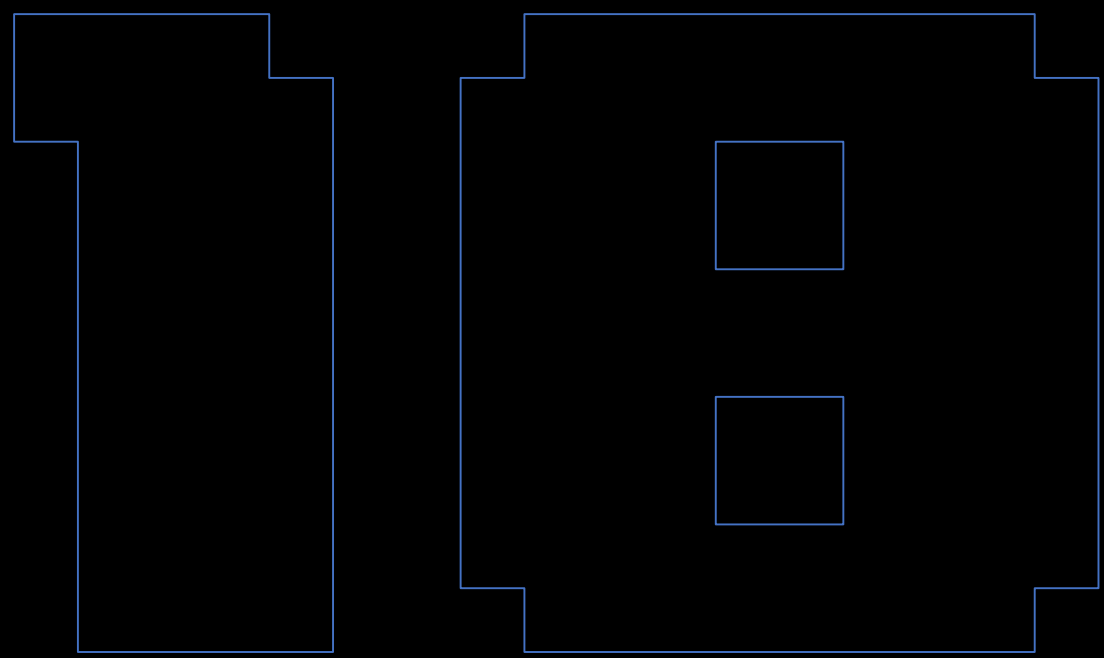
Além do KNN, existem muitos outros modelos prontos no scikit-learn. Cada um tem seus pontos fortes:

- KNN – Simples e direto
- Árvore de Decisão – Explicável, fácil de visualizar
- Random Forest – Conjunto de árvores, mais robusto
- SVM (Support Vector Machine) – Separação eficiente de classes
- MLP (Redes Neurais) – Ideal para dados complexos

Exemplo com Random Forest:

O FEITIÇO PYTHON - GUSTAVO.PY

```python
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
print("Acurácia:", model.score(X_test, y_test))
```



SALVANDO E CARREGANDO MODELOS

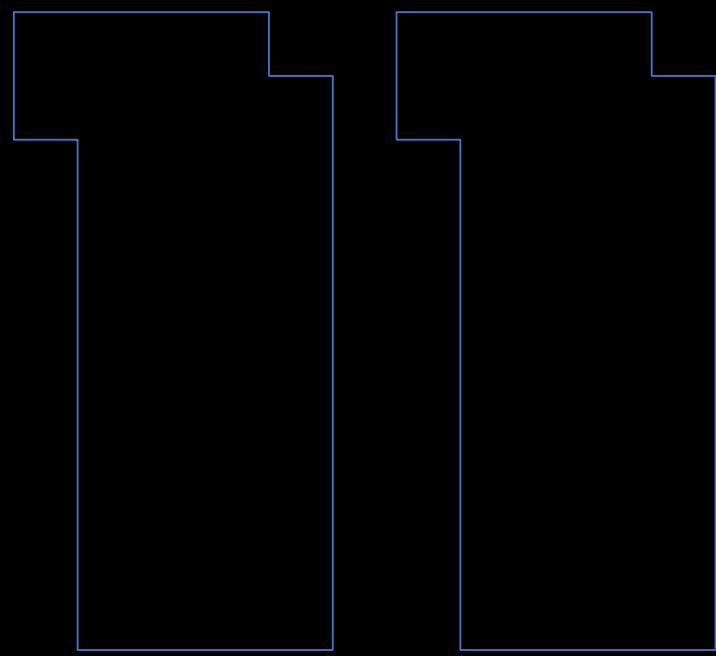
📦 SALVANDO E CARREGANDO MODELOS

Depois de treinar, você pode salvar o modelo e usá-lo depois sem precisar treinar de novo. Muito útil em sistemas reais.

Isso permite usar o modelo em APIs, aplicativos ou sites.

O FEITIÇO PYTHON - GUSTAVO.PY

```python
import joblib

joblib.dump(model, "modelo.pkl")              # Salvar
modelo_carregado = joblib.load("modelo.pkl") # Carregar
```



CONCLUSÕES

📌 CONCLUSÃO

Machine Learning não é mágica, mas também não é um bicho de sete cabeças. Com dados bem organizados e Python na mão, você já pode resolver problemas reais.
Comece com projetos simples:

- Diagnóstico de doenças
- Previsão de vendas
- Classificação de e-mails
- Análise de sentimentos

A prática é o que faz você evoluir. E lembre-se: entender os dados é mais importante que decorar algoritmos.

OBRIGADO POR LER ATÉ AQUI!

Este Ebook foi gerado por IA e diagramado por um humano.

-

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana ao conteúdo e pode conter erros gerados por uma IA.

https://github.com/GustavoCPadilha