Atividade 5 - Aprendizado de Máquina

AUTHOR
Rennan Dalla Guimarães

Exercício 1

1. Importar pacotes

```
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd

from sklearn.datasets import load_breast_cancer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedKFold, GridSearchCV

from sklearn.svm import SVC
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier
```

2. Carregar dados

```
data = load_breast_cancer()
X, y = data.data, data.target
feature_names = data.feature_names
print(f"{X.shape[0]} amostras · {X.shape[1]} atributos")
```

```
569 amostras · 30 atributos
```

3. Configurar validação cruzada estratificada

```
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

4. Definir modelos + grades de hiperparâmetros

```
results = []

# SVM ------------------------------------------------------------
svm_pipe = Pipeline([("scaler", StandardScaler()), ("clf", SVC())])
svm_param_grid = {
    "clf__kernel": ["linear", "rbf", "poly"],
    "clf__C": [0.1, 1, 10],
    "clf__gamma": ["scale", "auto"],
}
```

```
svm_grid = GridSearchCV(
    svm_pipe, svm_param_grid, cv=cv, scoring="accuracy", n_jobs=-1
)
svm_grid.fit(X, y)
results.append(
    {
        "Modelo": "SVM",
        "Acurácia_CV": svm_grid.best_score_,
        "Melhores_Params": svm_grid.best_params_,
    }
)

# Perceptron ----------------------------------------------------------
perc_pipe = Pipeline([("scaler", StandardScaler()),
                      ("clf", Perceptron(max_iter=1000, tol=1e-3))])
perc_param_grid = {
    "clf__penalty": [None, "l2", "l1", "elasticnet"],
    "clf__alpha": [1e-4, 1e-3, 1e-2],
    "clf__eta0": [0.1, 1.0, 10],
    "clf__fit_intercept": [True, False],
}
perc_grid = GridSearchCV(
    perc_pipe, perc_param_grid, cv=cv, scoring="accuracy", n_jobs=-1
)
perc_grid.fit(X, y)
results.append(
    {
        "Modelo": "Perceptron",
        "Acurácia_CV": perc_grid.best_score_,
        "Melhores_Params": perc_grid.best_params_,
    }
)

# MLP -----------------------------------------------------------------
mlp_pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", MLPClassifier(max_iter=500, random_state=42, early_stoppin
])
mlp_param_grid = {
    "clf__hidden_layer_sizes": [(50,), (100,), (50, 50)],
    "clf__activation": ["relu", "tanh"],
    "clf__alpha": [1e-4, 1e-3],
    "clf__learning_rate_init": [0.001, 0.01],
}
mlp_grid = GridSearchCV(
    mlp_pipe, mlp_param_grid, cv=cv, scoring="accuracy", n_jobs=-1
)
mlp_grid.fit(X, y)
results.append(
    {
        "Modelo": "MLP",
        "Acurácia_CV": mlp_grid.best_score_,
        "Melhores_Params": mlp_grid.best_params_,
    }
)

df_results = pd.DataFrame(results).set_index("Modelo")
df_results
```

Modelo	Acurácia_CV	Melhores_Params
SVM	0.978916	{'clf__C': 10, 'clf__gamma': 'scale', 'clf__ke...
Perceptron	0.971899	{'clf__alpha': 0.0001, 'clf__eta0': 0.1, 'clf_...
MLP	0.971899	{'clf__activation': 'tanh', 'clf__alpha': 0.00...

Exercício 2

1. Dataset codificado

```
import numpy as np

pacientes = {
    "João":  ([1, 1, 0, 1], 1),
    "Pedro": ([0, 0, 1, 0], 0),
    "Maria": ([1, 1, 0, 0], 0),
    "José":  ([1, 0, 1, 1], 1),
    "Ana":   ([1, 0, 0, 1], 0),
    "Leila": ([0, 0, 1, 1], 1),
}

X = np.array([v[0] for v in pacientes.values()], dtype=int)
d = np.array([v[1] for v in pacientes.values()], dtype=int)
```

2. Treinamento com log detalhado

```
eta = 0.5
threshold = -0.5
max_epochs = 100

w = np.zeros(X.shape[1])
b = 0.0

for epoch in range(max_epochs):
    print(f"\n===== Época {epoch} =====")
    errors = 0
    for idx, (xi, target) in enumerate(zip(X, d)):
        v = np.dot(w, xi) + b
        y = 1 if v >= threshold else 0
        erro = target - y
        print(f"Amostra {idx+1}: x={xi}, d={target}, v={v:.2f}, y={y},
        if erro != 0:
            old_w = w.copy()
            old_b = b
            w += eta * erro * xi
            b += eta * erro
            errors += 1
            print(f"  -> Ajuste! w: {old_w} -> {w},  b: {old_b:.2f} ->
        else:
```

```
            print("  -> Nenhum ajuste necessário.")
    print(f"Erros nesta época: {errors}")
    if errors == 0:
        print("Rede convergiu!")
        break

print("\nPesos finais:", w)
print("Bias final:", b)
```

```
===== Época 0 =====
Amostra 1: x=[1 1 0 1], d=1, v=0.00, y=1, erro=0
  -> Nenhum ajuste necessário.
Amostra 2: x=[0 0 1 0], d=0, v=0.00, y=1, erro=-1
  -> Ajuste! w: [0. 0. 0. 0.] -> [ 0.   0.  -0.5  0. ],  b: 0.00 ->
-0.50
Amostra 3: x=[1 1 0 0], d=0, v=-0.50, y=1, erro=-1
  -> Ajuste! w: [ 0.   0.  -0.5  0. ] -> [-0.5 -0.5 -0.5  0. ],  b:
-0.50 -> -1.00
Amostra 4: x=[1 0 1 1], d=1, v=-2.00, y=0, erro=1
  -> Ajuste! w: [-0.5 -0.5 -0.5  0. ] -> [ 0.  -0.5  0.   0.5],  b:
-1.00 -> -0.50
Amostra 5: x=[1 0 0 1], d=0, v=0.00, y=1, erro=-1
  -> Ajuste! w: [ 0.  -0.5  0.   0.5] -> [-0.5 -0.5  0.   0. ],  b:
-0.50 -> -1.00
Amostra 6: x=[0 0 1 1], d=1, v=-1.00, y=0, erro=1
  -> Ajuste! w: [-0.5 -0.5  0.   0. ] -> [-0.5 -0.5  0.5  0.5],  b:
-1.00 -> -0.50
Erros nesta época: 5

===== Época 1 =====
Amostra 1: x=[1 1 0 1], d=1, v=-1.00, y=0, erro=1
  -> Ajuste! w: [-0.5 -0.5  0.5  0.5] -> [0.  0.  0.5 1. ],  b: -0.50
-> 0.00
Amostra 2: x=[0 0 1 0], d=0, v=0.50, y=1, erro=-1
  -> Ajuste! w: [0.  0.  0.5 1. ] -> [0. 0. 0. 1.],  b: 0.00 -> -0.50
Amostra 3: x=[1 1 0 0], d=0, v=-0.50, y=1, erro=-1
  -> Ajuste! w: [0. 0. 0. 1.] -> [-0.5 -0.5  0.   1. ],  b: -0.50 ->
-1.00
Amostra 4: x=[1 0 1 1], d=1, v=-0.50, y=1, erro=0
  -> Nenhum ajuste necessário.
Amostra 5: x=[1 0 0 1], d=0, v=-0.50, y=1, erro=-1
  -> Ajuste! w: [-0.5 -0.5  0.   1. ] -> [-1.  -0.5  0.   0.5],  b:
-1.00 -> -1.50
Amostra 6: x=[0 0 1 1], d=1, v=-1.00, y=0, erro=1
  -> Ajuste! w: [-1.  -0.5  0.   0.5] -> [-1.  -0.5  0.5  1. ],  b:
-1.50 -> -1.00
Erros nesta época: 5

===== Época 2 =====
Amostra 1: x=[1 1 0 1], d=1, v=-1.50, y=0, erro=1
  -> Ajuste! w: [-1.  -0.5  0.5  1. ] -> [-0.5  0.   0.5  1.5],  b:
-1.00 -> -0.50
Amostra 2: x=[0 0 1 0], d=0, v=0.00, y=1, erro=-1
  -> Ajuste! w: [-0.5  0.   0.5  1.5] -> [-0.5  0.   0.   1.5],  b:
-0.50 -> -1.00
Amostra 3: x=[1 1 0 0], d=0, v=-1.50, y=0, erro=0
  -> Nenhum ajuste necessário.
```

```
Amostra 4: x=[1 0 1 1], d=1, v=0.00, y=1, erro=0
  -> Nenhum ajuste necessário.
Amostra 5: x=[1 0 0 1], d=0, v=0.00, y=1, erro=-1
  -> Ajuste! w: [-0.5  0.   0.   1.5] -> [-1.  0.  0.  1.],  b: -1.00
-> -1.50
Amostra 6: x=[0 0 1 1], d=1, v=-0.50, y=1, erro=0
  -> Nenhum ajuste necessário.
Erros nesta época: 3

===== Época 3 =====
Amostra 1: x=[1 1 0 1], d=1, v=-1.50, y=0, erro=1
  -> Ajuste! w: [-1.  0.  0.  1.] -> [-0.5  0.5  0.   1.5],  b: -1.50
-> -1.00
Amostra 2: x=[0 0 1 0], d=0, v=-1.00, y=0, erro=0
  -> Nenhum ajuste necessário.
Amostra 3: x=[1 1 0 0], d=0, v=-1.00, y=0, erro=0
  -> Nenhum ajuste necessário.
Amostra 4: x=[1 0 1 1], d=1, v=0.00, y=1, erro=0
  -> Nenhum ajuste necessário.
Amostra 5: x=[1 0 0 1], d=0, v=0.00, y=1, erro=-1
  -> Ajuste! w: [-0.5  0.5  0.   1.5] -> [-1.   0.5  0.   1. ],  b:
-1.00 -> -1.50
Amostra 6: x=[0 0 1 1], d=1, v=-0.50, y=1, erro=0
  -> Nenhum ajuste necessário.
Erros nesta época: 2

===== Época 4 =====
Amostra 1: x=[1 1 0 1], d=1, v=-1.00, y=0, erro=1
  -> Ajuste! w: [-1.   0.5  0.   1. ] -> [-0.5  1.   0.   1.5],  b:
-1.50 -> -1.00
Amostra 2: x=[0 0 1 0], d=0, v=-1.00, y=0, erro=0
  -> Nenhum ajuste necessário.
Amostra 3: x=[1 1 0 0], d=0, v=-0.50, y=1, erro=-1
  -> Ajuste! w: [-0.5  1.   0.   1.5] -> [-1.   0.5  0.   1.5],  b:
-1.00 -> -1.50
Amostra 4: x=[1 0 1 1], d=1, v=-1.00, y=0, erro=1
  -> Ajuste! w: [-1.   0.5  0.   1.5] -> [-0.5  0.5  0.5  2. ],  b:
-1.50 -> -1.00
Amostra 5: x=[1 0 0 1], d=0, v=0.50, y=1, erro=-1
  -> Ajuste! w: [-0.5  0.5  0.5  2. ] -> [-1.   0.5  0.5  1.5],  b:
-1.00 -> -1.50
Amostra 6: x=[0 0 1 1], d=1, v=0.50, y=1, erro=0
  -> Nenhum ajuste necessário.
Erros nesta época: 4

===== Época 5 =====
Amostra 1: x=[1 1 0 1], d=1, v=-0.50, y=1, erro=0
  -> Nenhum ajuste necessário.
Amostra 2: x=[0 0 1 0], d=0, v=-1.00, y=0, erro=0
  -> Nenhum ajuste necessário.
Amostra 3: x=[1 1 0 0], d=0, v=-2.00, y=0, erro=0
  -> Nenhum ajuste necessário.
Amostra 4: x=[1 0 1 1], d=1, v=-0.50, y=1, erro=0
  -> Nenhum ajuste necessário.
Amostra 5: x=[1 0 0 1], d=0, v=-1.00, y=0, erro=0
  -> Nenhum ajuste necessário.
Amostra 6: x=[0 0 1 1], d=1, v=0.50, y=1, erro=0
  -> Nenhum ajuste necessário.
```

```
Erros nesta época: 0
Rede convergiu!

Pesos finais: [-1.   0.5  0.5  1.5]
Bias final: -1.5
```

3. Teste com novos pacientes

```python
def classificar(xi):
    v = np.dot(w, xi) + b
    return 1 if v >= threshold else 0

testes = {
    "Luis":  np.array([0, 0, 0, 1]),
    "Laura": np.array([1, 1, 1, 1]),
}

print("\n=== Testes finais ===")
for nome, xi in testes.items():
    pred = classificar(xi)
    print(f"{nome}: {'Doente' if pred else 'Saudável'}")
```

```
=== Testes finais ===
Luis: Doente
Laura: Doente
```