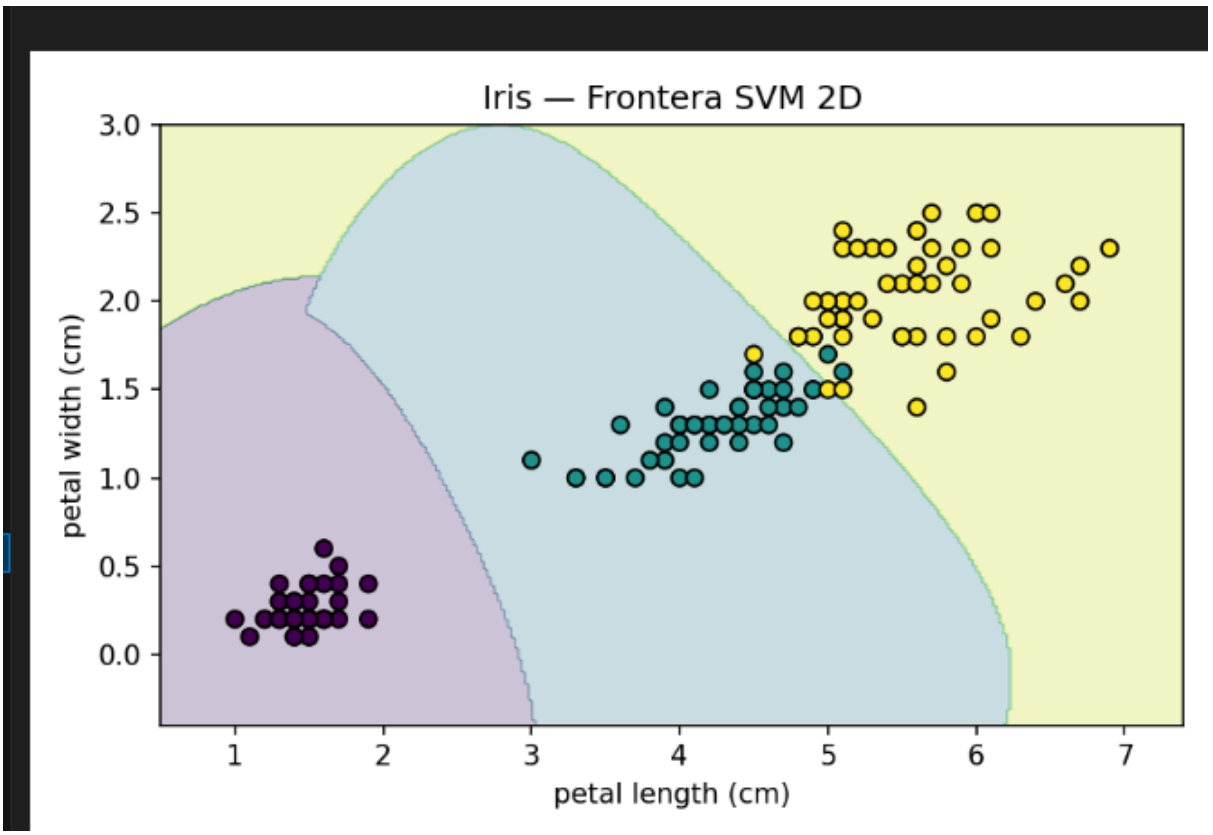
Guillermo García Hernández
2º DAW 25/26
IA

Introducción al ML (SVM)

```python
from sklearn import datasets, svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import numpy as np

yo = "Guillermo García Hernández - 2º DAW"


# 1) Cambiar kernel y observar diferencias
iris = datasets.load_iris()
X, y = iris.data, iris.target
X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.25, stratify=y, random_state=42)

resultados_k = []
for kernel in ["linear", "rbf", "poly"]:
    clf = svm.SVC(kernel=kernel, gamma="scale")
    clf.fit(X_tr, y_tr)
    acc = accuracy_score(y_te, clf.predict(X_te))
    resultados_k.append((kernel, acc))

print("EJERCICIO 1 (Iris)  Cambiar kernel")
for k, a in resultados_k:
    print(f"Kernel={k:6s} → Precisión={a:.3f}")


mejor_k = max(resultados_k, key=lambda t: t[1])[0]
peor_k  = min(resultados_k, key=lambda t: t[1])[0]
print(
    f"- Probé 'linear', 'rbf' y 'poly'. En mi caso el mejor fue '{mejor_k}' y el más flojo '{peor_k}'.\n"
    "- En Iris las clases están bastante separadas; por eso 'linear' y 'rbf' suelen ir muy bien.\n"
    "- 'poly' a veces rinde un poco menos porque puede sobreajustar.\n"
)
print(yo)
```

```python
# 2) Cambiar test_size
print("EJERCICIO 2 (Iris) Cambiar test_size (0.2, 0.3, 0.4)")
resultados_ts = []
for ts in [0.2, 0.3, 0.4]:
    X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=ts,
    stratify=y, random_state=42)
    clf = svm.SVC(kernel="rbf", gamma="scale")
    clf.fit(X_tr, y_tr)
    acc = accuracy_score(y_te, clf.predict(X_te))
    resultados_ts.append((ts, acc))
    print(f"test_size={ts} → Precisión={acc:.3f}")

accs = [a for _, a in resultados_ts]
varia = max(accs) - min(accs)
print(
    f"- Comparé 0.2, 0.3 y 0.4 de test. La variación de precisión fue de ~
    {varia:.3f}.\n"
    "- Al subir test_size hay menos datos para entrenar; por eso puede
    bajar un poco la precisión.\n"
    "- En Iris el efecto es pequeño porque el problema es fácil de separar.
    \n"
)
print(yo)
```

```python
# 3) Frontera 2D cambiando a dos características
print("EJERCICIO 3 (Iris) Frontera 2D con dos características")
f1, f2 = 2, 3
X2 = X[:, [f1, f2]]
X2_tr, X2_te, y2_tr, y2_te = train_test_split(X2, y, test_size=0.25, stratify=y, random_state=42)

clf2d = svm.SVC(kernel="rbf", gamma="auto")
clf2d.fit(X2_tr, y2_tr)

# malla para la frontera
x_min, x_max = X2[:, 0].min() - 0.5, X2[:, 0].max() + 0.5
y_min, y_max = X2[:, 1].min() - 0.5, X2[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300), np.linspace(y_min, y_max, 300))
Z = clf2d.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

plt.figure(figsize=(6, 4))
plt.contourf(xx, yy, Z, alpha=0.25)
plt.scatter(X2[:, 0], X2[:, 1], c=y, edgecolors="k")
plt.title("Iris — Frontera SVM 2D")
plt.xlabel(iris.feature_names[f1])
plt.ylabel(iris.feature_names[f2])
plt.tight_layout()
plt.savefig("iris_frontera_2d.png", dpi=150)
plt.close()

print(
    f"- Cambié a 2 características para dibujar el plano: {iris.feature_names[f1]} vs {iris.feature_names[f2]}.\n"
    "- Con las variables de pétalo (2 y 3) la separación se ve muy limpia; con sépalo suele mezclarse más.\n"
)
print(yo)
```


Iris — Frontera SVM 2D
petal width (cm)
petal length (cm)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7

```python
# CÁNCER DE MAMA

print("EJERCICIO 4 (Cáncer) Cambiar C (0.1, 1, 10)")
cancer = datasets.load_breast_cancer()
Xc, yc = cancer.data, cancer.target
Xc_tr, Xc_te, yc_tr, yc_te = train_test_split(Xc, yc, test_size=0.25, stratify=yc, random_state=42)

accs_C = []
for C in [0.1, 1, 10]:
    clf = svm.SVC(kernel="rbf", C=C, gamma="scale")
    clf.fit(Xc_tr, yc_tr)
    y_pred = clf.predict(Xc_te)
    acc = accuracy_score(yc_te, y_pred)
    accs_C.append((C, acc))
    print(f"\nC = {C} ")
    print(classification_report(yc_te, y_pred, target_names=cancer.target_names))
    print(f"C={C}: {acc:.3f}")


mejor_C = max(accs_C, key=lambda t: t[1])[0]
print(
    f"- Probé C=0.1, 1 y 10. El mejor lo obtuve con C={mejor_C}.\n"
    "- C pequeño (0.1): margen más grande, más generalización, puede bajar recall/precision.\n"
    "- C grande (10): margen pequeño, se ajusta más y puede subir el rendimiento, pero arriesga sobreajuste.\n"
)
print(yo)
```

```python
print("EJERCICIO 5 (Cáncer) Aciertos / Fallos")
clf = svm.SVC(kernel="rbf", C=1, gamma="scale")
clf.fit(Xc_tr, yc_tr)
yc_pred = clf.predict(Xc_te)
aciertos = (yc_pred == yc_te).sum()
fallos   = (yc_pred != yc_te).sum()
print(f"Aciertos totales: {aciertos} | Fallos totales: {fallos}")

print(
    "- Estos totales confirman el rendimiento global; conviene mirarlos junto al informe de clasificación\n"
    "  para ver el equilibrio entre precision y recall en cada clase.\n"
)
print(yo)
```

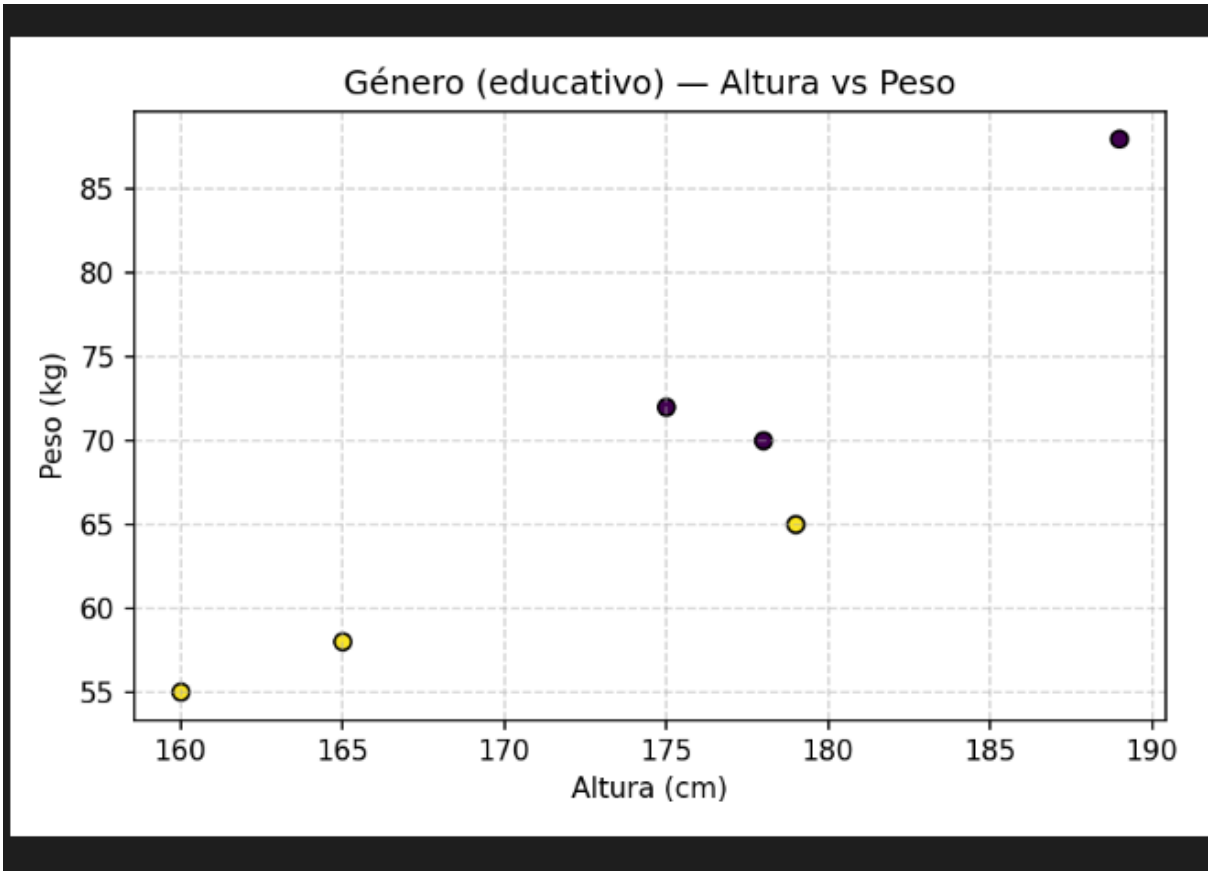
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
Guillermo García Hernández - 2º DAW
EJERCICIO 5 (Cáncer) Aciertos / Fallos
Aciertos totales: 132 | Fallos totales: 11
- Estos totales confirman el rendimiento global; conviene mirarlos junto al informe de clas
ificación
  para ver el equilibrio entre precision y recall en cada clase.
```

```python
# GÉNERO

print("EJERCICIO 6 Añadir ejemplos y reentrenar")
# Datos: [altura, peso, talla]; etiquetas: 0=Masculino, 1=Femenino
Xg = [[178,70,9],[189,88,11],[179,65,8],[160,55,7]]
yg = [0,0,1,1]
clf_g = svm.SVC(kernel="linear")
clf_g.fit(Xg, yg)

# añadir y reentrenar
Xg.extend([[175,72,9],[165,58,7]])
yg.extend([0,1])
clf_g.fit(Xg, yg)
print(f"→ Reentrenado con {len(Xg)} ejemplos.")
```

```
print("EJERCICIO 7  Predicciones puntuales")
casos = [[185,80,10],[162,55,7]]
for c in casos:
    pred = clf_g.predict([c])[0]
    print(f"Entrada {c} → Predicción: {'Masculino' if pred==0 else
    'Femenino'}")

print(
    "- La 'frontera' intuitiva separa combinaciones de mayor altura/peso/
    talla (tiende a 0) de las menores (tiende a 1).\n"
)
print(yo)
```

```
print("EJERCICIO 8 Altura vs Peso coloreado por clase")
X_arr = np.array(Xg); y_arr = np.array(yg)
plt.figure(figsize=(6,4))
plt.scatter(X_arr[:,0], X_arr[:,1], c=y_arr, edgecolors="k")
plt.xlabel("Altura (cm)"); plt.ylabel("Peso (kg)")
plt.title("Género (educativo) — Altura vs Peso")
plt.grid(True, linestyle="--", alpha=0.5)
plt.tight_layout()
plt.savefig("genero_altura_peso.png", dpi=150)
plt.close()

print(
    "- El scatter muestra dos grupos aproximados, pero se ve que la
    separación es muy tosca.\n"
)
print(yo)
```



EJERCICIO 9 Discusión (límites y riesgos)
Conclusión: SVM es potente, pero debe aplicarse con responsabilidad.