## 

# librerias:

# Librerías de evaluación

from sklearn.metrics import precision\_score, recall\_score, f1\_score, accuracy\_score, confusion\_matrix, roc\_curve, auc

from sklearn.model\_selection import cross\_val\_score

## PRECISDION, RECALL, F1, ACCURACY

# Cálculo de las métricas

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

# Imprimir los resultados

print("Precisión:", precision)

print("Recall:", recall)

print("F1-Score:", f1)

print("Accuracy:", accuracy)

## MATRIZ CONFUSION

matriu\_confusion = confusion\_matrix(y\_test, y\_pred)

print("Matriz de Confusión:\n", matriu\_confusion)

## OVERFITITYNG UNDERFITING

# Fer prediccions sobre el conjunt d'entrenament

y\_train\_pred = best\_knn.predict(X\_train)

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

print(train\_accuracy)

# Fer prediccions sobre el conjunt de test

y\_test\_pred = best\_knn.predict(X\_test)

# Calcular la precisió sobre el conjunt de test

test\_accuracy = accuracy\_score(y\_test, y\_test\_pred)

print(test\_accuracy)

## cross validation

# Realitzar cross-validation (aquí fem 5 "folds") cv\_scores = cross\_val\_score(knn, X\_train\_scaled, y\_train, cv=5) # cv=5 indica 5 "folds" # Imprimir els resultats print("Puntuacions de cross-validation:", cv\_scores) print("Mitjana de les puntuacions:", cv\_scores.mean())

## CURVA ROC

# Calcular les probabilitats per la classe positiva per cada model

knn\_probs = best\_knn.predict\_proba(X\_test)[:, 1]

dt\_probs = dt\_model.predict\_proba(X\_test)[:, 1]

best\_dt\_probs = best\_dt\_model.predict\_proba(X\_test)[:, 1]

rf\_probs = best\_rf.predict\_proba(X\_test)[:, 1]

# Calcular les FPR i TPR per cada model

fpr\_knn, tpr\_knn, \_ = roc\_curve(y\_test, knn\_probs)

fpr\_dt, tpr\_dt, \_ = roc\_curve(y\_test, dt\_probs)

fpr\_best\_dt, tpr\_best\_dt, \_ = roc\_curve(y\_test, best\_dt\_probs)

fpr\_rf, tpr\_rf, \_ = roc\_curve(y\_test, rf\_probs)

# Calcular l'AUC per a cada model

roc\_auc\_knn = auc(fpr\_knn, tpr\_knn)

roc\_auc\_dt = auc(fpr\_dt, tpr\_dt)

roc\_auc\_best\_dt = auc(fpr\_best\_dt, tpr\_best\_dt)

roc\_auc\_rf = auc(fpr\_rf, tpr\_rf)

# Crear la gràfica de la corba ROC

plt.figure(figsize=(8, 6))

# Traçar les corbes ROC

plt.plot(fpr\_knn, tpr\_knn, label=f'KNN (AUC = {roc\_auc\_knn:.2f})')

plt.plot(fpr\_dt, tpr\_dt, label=f'Decision Tree (AUC = {roc\_auc\_dt:.2f})')

plt.plot(fpr\_best\_dt, tpr\_best\_dt, label=f'Optimized DT (AUC = {roc\_auc\_best\_dt:.2f})')

plt.plot(fpr\_rf, tpr\_rf, label=f'Random Forest (AUC = {roc\_auc\_rf:.2f})')

# Línia diagonal (classificador aleatori)

plt.plot([0, 1], [0, 1], 'k--')

# Títol i etiquetes

plt.title('Comparació de Models - Curva ROC')

plt.xlabel('Falsos Positius (FPR)')

plt.ylabel('Veritables Positius (TPR)')

plt.legend(loc='lower right')

# Mostrar el gràfic

plt.show()