Các câu so sánh hiện có :

1. So sánh hiệu năng giữa mô hình k-NN và mô hình Linear Regression sử dụng các

độ đo phù hợp (ví dụ: Accuracy cho K-NN và MSE cho Linear Regression).

2. So sánh hiệu năng giữa mô hình Decision Tree Classifier và Gradient Boosting Regression sử dụng các độ đo phù hợp.

3. So sánh hiệu năng giữa mô hình SVM và mô hình Random Forest Regression sử dụng các độ đo phù hợp

4.So sánh hiệu năng giữa mô hình Logistic Regression và mô hình Linear Regression sử dụng các độ đo hiệu năng phù hợp.

Code Python thử nghiệm :

|  |  |  |
| --- | --- | --- |
| Tên mô hình | Module tương ứng | Cách ghi nhớ |
| Linner Regression  Losgistic Regression | Linnear\_model | Hàm tuyến tính => linner |
| DecisionTreeClassifier | tree | Cây quyết định |
| RandomForest,GradientBoosting | ensemble | Tổ hợp nhiều mô hình |
| Svc,svr | svm | Svm |
| KneighborsClassifier | neighbor | Hàng xóm ( Học Lười ) |
| Kmeans | cluster | Phân cụm |

### ✅ 1. **Phân loại các nhóm chính trong** sklearn (nên nhớ):

| Module | Chứa những gì |
| --- | --- |
| sklearn.linear\_model | Linear Regression, Logistic Regression |
| sklearn.tree | DecisionTreeClassifier, DecisionTreeRegressor |
| sklearn.ensemble | Random Forest, Gradient Boosting, AdaBoost |
| sklearn.svm | SVM: SVC, SVR |
| sklearn.neighbors | KNN: KNeighborsClassifier, KNeighborsRegressor |
| sklearn.naive\_bayes | GaussianNB, MultinomialNB |
| sklearn.cluster | KMeans, DBSCAN, v.v. |
| sklearn.datasets | Bộ dữ liệu có sẵn |
| sklearn.model\_selection | train\_test\_split, cross\_val\_score |
| sklearn.metrics | accuracy\_score, mean\_squared\_error, confusion\_matrix |

Các loại bài tập

Bản Failse1:

Code bài 1 :

**Câu 1: 4.0 điểm**

Sử dụng bộ dữ liệu Iris có trong thư viện Scikit-Learn để thực hiện các bước sau:

1. Tải bộ dữ liệu Iris và hiển thị thông tin cơ bản về bộ dữ liệu này.

2. Chia bộ dữ liệu thành tập huấn luyện và tập đánh giá hiệu năng của mô hình.

3. Áp dụng mô hình K-Nearest Neighbors (k-NN) từ thư viện Scikit-Learn để huấn

luyện trên tập dữ liệu huấn luyện.

4. Đánh giá hiệu năng của mô hình đã huấn luyện dựa trên bộ dữ liệu đánh giá, sử

dụng độ đo chính xác (Accuracy).

5. Vẽ biểu đồ ma trận nhầm lẫn (Confusion Matrix) cho kết quả dự đoán trên tập

kiểm tra.

from sklearn.datasets import load\_iris

# tai du lieu

iris = load\_iris()

# chuyen sang DataFrame

import pandas as pd

df = pd.DataFrame(data=iris.data,columns=iris.feature\_names)

df['target']=iris.target

#hien thi thong tin

print(df.head())

print(df.describe())

print(iris.target\_names)

# chia tap du lieu

from sklearn.model\_selection import train\_test\_split

x= df[iris.feature\_names]

y= df['target']

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=42)

# huấn luyện bằng K-NN

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(x\_train,y\_train)

# Dự đoán dánh giá mô hình

from sklearn.metrics import accuracy\_score

y\_pred=knn.predict(x\_test)

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

print("Độ chính xác của mô hình là: ",accuracy)

# Vẽ biểu đồ ma trânj nhầm lẫn  Confusion\_matrix

from sklearn.metrics import confusion\_matrix,ConfusionMatrixDisplay

import matplotlib.pyplot as plt

cm = confusion\_matrix(y\_test,y\_pred)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm,display\_labels=iris.target\_names)

disp.plot(cmap='pink')

plt.title("confusion\_matrix -iris - knn ")

plt.show()

**Câu 2: 4.0 điểm**

Sử dụng bộ dữ liệu Boston Housing (có sẵn trong thư viện sklearn.datasets) để thực hiện các bước sau:

1. Tải bộ dữ liệu Boston Housing và hiển thị thông tin cơ bản về bộ dữ liệu này.

2. Chia bộ dữ liệu Boston Housing thành tập dữ liệu huấn luyện và tập dữ liệu đánh

giá hiệu năng mô hình.

3. Áp dụng mô hình Linear Regression từ thư viện Scikit-Learn để huấn luyện trên

tập huấn luyện.

4. Đánh giá hiệu năng của mô hình trên tập dữ liệu đánh giá bằng cách tính toán sai

số trung bình bình phương (Mean Squared Error – MSE).

5. Vẽ biểu đồ scatter plot so sánh giá trị dự đoán và giá trị thực tế trên tập kiểm tra.

from sklearn.datasets import load\_boston

# tai du lieu ve

boston = load\_boston()

# Đưa vào DataFrame

import pandas as pd

df=pd.DataFrame(data=boston.data,columns=boston.feature\_names)

df['PRICE']=boston.target

# hien thi thong tin co ban ve bo du lieu

print(df.head())

print(df.describe())

# chia bo du lieu

from sklearn.model\_selection import train\_test\_split

x=df.drop('PRICE',axis=1)

y=df['PRICE']

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=42)

# huan  luyen bang LinerREgression

from sklearn.linear\_model import LinearRegression

model\_ir = LinearRegression()

model\_ir.fit(x\_train,y\_train)

# danh gia hieu nang baang MSE (Mean Squared Error)

from sklearn.metrics import mean\_squared\_error

y\_pred = model\_ir.predict(x\_test)

mea = mean\_squared\_error(y\_test,y\_pred)

print("DO chinh xac cua bo du lieu la: ",mea)

#. Vẽ biểu đồ scatter plot so sánh giá trị dự đoán và giá trị thực tế trên tập kiểm tra.

import matplotlib.pyplot as plt

plt.scatter(y\_test,y\_pred)

plt.xlabel('gia tri thuc te ')

plt.ylabel('gia tri du doan')

plt.plot([y\_test.min(),y\_test.max()],[y\_test.min(),y\_test.max()],'r--')

plt.show()

**Câu 3: 2.0 điểm**

1. So sánh hiệu năng giữa mô hình k-NN và mô hình Linear Regression sử dụng các độ đo phù hợp (ví dụ: Accuracy cho K-NN và MSE cho Linear Regression).

2. Đề xuất một phương pháp cải thiện hiệu năng của mô hình phân lớp hoặc mô hình

hồi quy dựa trên các kết quả thực hiện được trong phần 1 và phần 2.

# ==== CÂU 3: SO SÁNH HIỆU NĂNG & CẢI THIỆN MÔ HÌNH ====

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy\_score, mean\_squared\_error

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import RidgeCV

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_iris, load\_boston

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LinearRegression

# -----------------------------

# PHẦN 1: Huấn luyện lại mô hình Câu 1 (k-NN trên Iris)

# -----------------------------

iris = load\_iris()

X\_iris = iris.data

y\_iris = iris.target

X\_train\_iris, X\_test\_iris, y\_train\_iris, y\_test\_iris = train\_test\_split(X\_iris, y\_iris, test\_size=0.2, random\_state=42)

knn\_model = KNeighborsClassifier()

knn\_model.fit(X\_train\_iris, y\_train\_iris)

y\_pred\_knn = knn\_model.predict(X\_test\_iris)

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

print(f"Độ chính xác (Accuracy) của mô hình k-NN trên tập Iris: {accuracy\_knn:.4f}")

# -----------------------------

# PHẦN 2: Huấn luyện lại mô hình Câu 2 (Linear Regression trên Boston)

# -----------------------------

boston = load\_boston()

X\_boston = boston.data

y\_boston = boston.target

X\_train\_boston, X\_test\_boston, y\_train\_boston, y\_test\_boston = train\_test\_split(X\_boston, y\_boston, test\_size=0.2, random\_state=42)

lr\_model = LinearRegression()

lr\_model.fit(X\_train\_boston, y\_train\_boston)

y\_pred\_lr = lr\_model.predict(X\_test\_boston)

mse\_lr = mean\_squared\_error(y\_test\_boston, y\_pred\_lr)

print(f"MSE của mô hình Linear Regression trên tập Boston: {mse\_lr:.4f}")

# -----------------------------

# PHẦN 3: Vẽ biểu đồ so sánh hiệu năng 2 mô hình

# -----------------------------

plt.figure(figsize=(8,5))

plt.bar(['k-NN (Accuracy)', 'Linear Reg (1 / (1 + MSE))'],

        [accuracy\_knn, 1/(1 + mse\_lr)], color=['lightgreen', 'lightskyblue'])

plt.title('So sánh hiệu năng: k-NN vs Linear Regression')

plt.ylabel('Chỉ số hiệu năng (Càng cao càng tốt)')

plt.grid(True)

plt.show()

# -----------------------------

# PHẦN 4: CẢI THIỆN MÔ HÌNH PHÂN LỚP - RandomForest

# -----------------------------

print("\n--- CẢI THIỆN MÔ HÌNH PHÂN LỚP (Iris) ---")

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train\_iris, y\_train\_iris)

y\_pred\_rf = rf\_model.predict(X\_test\_iris)

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

print(f"Accuracy sau khi cải thiện (RandomForest): {accuracy\_rf:.4f}")

# -----------------------------

# PHẦN 5: CẢI THIỆN MÔ HÌNH HỒI QUY - RidgeCV

# -----------------------------

print("\n--- CẢI THIỆN MÔ HÌNH HỒI QUY (Boston) ---")

ridge\_model = RidgeCV(alphas=[0.1, 1.0, 10.0])

ridge\_model.fit(X\_train\_boston, y\_train\_boston)

y\_pred\_ridge = ridge\_model.predict(X\_test\_boston)

mse\_ridge = mean\_squared\_error(y\_test\_boston, y\_pred\_ridge)

print(f"MSE sau khi cải thiện (RidgeCV): {mse\_ridge:.4f}")

# -----------------------------

# PHẦN 6: Biểu đồ so sánh trước/sau cải thiện

# -----------------------------

plt.figure(figsize=(8,5))

models = ['k-NN', 'RandomForest', 'LinearReg', 'RidgeCV']

scores = [accuracy\_knn, accuracy\_rf, 1/(1 + mse\_lr), 1/(1 + mse\_ridge)]

colors = ['gray', 'green', 'orange', 'blue']

plt.bar(models, scores, color=colors)

plt.title("Hiệu năng mô hình trước và sau khi cải thiện")

plt.ylabel("Chỉ số hiệu năng (Accuracy / 1 / (1 + MSE))")

plt.grid(True)

plt.show()

**Đề 2 :**

**Câu 1 :**

**Sử dụng bộ dữ liệu Wine trong thư viện Scikit-Learn để thực hiện các bước sau:**

**1.Tải bộ dữ liệu Wine và hiển thị thông tin cơ bản về bộ dữ liệu này.**

**2.Chia bộ dữ liệu thành tập huấn luyện và tập đánh giá hiệu năng của mô hình.**

**3.Áp dụng mô hình Decision Tree Classifier từ thư viện Scikit-Learn để huấn luyện trên tập dữ liệu huấn luyện.**

**4.Đánh giá hiệu năng của mô hình đã huấn luyện dựa trên bộ dữ liệu đánh giá, sử dụng độ đo chính xác (Accuracy).**

**5.Vẽ biểu đồ cây quyết định của mô hình đã huấn luyện.**

**################**

**Ý 2 thật ra cũng hơi khác chút cho vào cho an toàn :**

# huan luyen bang mo hinh Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

dtc.fit(x\_train,y\_train)

**Ý 1,3,4 làm bth ý 5 thì cây quyết định kiểu nó khác :**

# Ve bieu do cay quyet dinh cua mo hinh huan luyen

from sklearn.tree import plot\_tree as pl

import matplotlib.pyplot as plt

plt.figure(figsize=(20,10))

pl(dtc,feature\_names=wine.feature\_names,class\_names=wine.target\_names,filled=True)

plt.title(" Tree\_matrrix-wine-DTreeCL ")

plt.show()

**Câu 2: 4.0 điểm**

**Sử dụng bộ dữ liệu California Housing trong thư viện sklearn.datasets để thực hiện các bước sau:**

**1. Tải bộ dữ liệu California Housing và hiển thị thông tin cơ bản về bộ dữ liệu này.**

**2. Chia bộ dữ liệu California Housing thành tập dữ liệu huấn luyện và tập dữ liệu đánh giá hiệu năng mô hình.**

**3. Áp dụng mô hình Gradient Boosting Regression từ thư viện Scikit-Learn để huấn luyện trên tập huấn luyện.**

**4. Đánh giá mô hình dựa trên tập dữ liệu đánh giá hiệu năng của mô hình bằng cách tính toán sai số trung bình bình phương (Mean Squared Error – MSE).**

**5. Vẽ biểu đồ scatter plot so sánh giá trị dự đoán và giá trị thực tế trên tập kiểm tra.**

from sklearn.datasets import fetch\_california\_housing

CH = fetch\_california\_housing()

#Dua vao DataFrame

import pandas as pd

df = pd.DataFrame(data=CH.data,columns=CH.feature\_names)

df['PRICE']=CH.target

# Hien thi thong tin co ban ve bo du lieu

print(df.head())

print(df.describe())

# chia bo du lieu

from sklearn.model\_selection import train\_test\_split

x=df.drop('PRICE',axis=1)

y=df['PRICE']

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=42)

# Huan luyen bang Gradiemt Boosting Regression

from sklearn.ensemble import GradientBoostingRegressor

model\_ch=GradientBoostingRegressor()

model\_ch.fit(x\_train,y\_train)

#Danh gia hieu nang bàng MSE

from sklearn.metrics import mean\_absolute\_error

y\_pred = model\_ch.predict(x\_test)

mea = mean\_absolute\_error(y\_test,y\_pred)

print("Do chinh xac cua mo hinh la : ",mea)

# ve bieu do scatter plot so sanh gia tri du doan voi gia tri thuc te

import matplotlib.pyplot as plt

plt.scatter(y\_test,y\_pred)

plt.xlabel('gia tri thuc te: ')

plt.ylabel('gia tri du doan ')

plt.plot([y\_test.min(),y\_test.max()],[y\_test.min(),y\_test.max()],'r--')

plt.show()

**Câu 3: 2.0 điểm**

**1. So sánh hiệu năng giữa mô hình Decision Tree Classifier và Gradient Boosting Regression sử dụng các độ đo phù hợp.**

**2. Đề xuất một phương pháp cải thiện hiệu năng của mô hình phân lớp hoặc mô hình hồi quy dựa trên các kết quả thực hiện được trong phần 1 và phần 2.**

# # ============================ CÂU 3: SO SÁNH HIỆU NĂNG ============================

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_wine, fetch\_california\_housing

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingRegressor

from sklearn.metrics import accuracy\_score, mean\_squared\_error

# --- DỮ LIỆU WINE ---

wine = load\_wine()

df\_wine = pd.DataFrame(wine.data, columns=wine.feature\_names)

df\_wine['target'] = wine.target

x\_wine = df\_wine[wine.feature\_names]

y\_wine = df\_wine['target']

x\_train\_wine, x\_test\_wine, y\_train\_wine, y\_test\_wine = train\_test\_split(x\_wine, y\_wine, test\_size=0.2, random\_state=42)

# Huấn luyện Decision Tree

dtc = DecisionTreeClassifier()

dtc.fit(x\_train\_wine, y\_train\_wine)

y\_pred\_wine = dtc.predict(x\_test\_wine)

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

print(f"Accuracy mô hình DecisionTreeClassifier: {accuracy\_wine:.4f}")

# --- DỮ LIỆU CALIFORNIA ---

cal = fetch\_california\_housing()

df\_cal = pd.DataFrame(cal.data, columns=cal.feature\_names)

df\_cal['PRICE'] = cal.target

x\_cal = df\_cal.drop('PRICE', axis=1)

y\_cal = df\_cal['PRICE']

x\_train\_cal, x\_test\_cal, y\_train\_cal, y\_test\_cal = train\_test\_split(x\_cal, y\_cal, test\_size=0.2, random\_state=42)

# Huấn luyện Gradient Boosting

gbr = GradientBoostingRegressor()

gbr.fit(x\_train\_cal, y\_train\_cal)

y\_pred\_cal = gbr.predict(x\_test\_cal)

mse\_cal = mean\_squared\_error(y\_test\_cal, y\_pred\_cal)

print(f"MSE mô hình GradientBoostingRegressor: {mse\_cal:.4f}")

# --- VẼ BIỂU ĐỒ SO SÁNH ---

plt.figure(figsize=(8,5))

plt.bar(['Decision Tree\n(Accuracy)', 'Gradient Boosting\n(1/(1+MSE))'],

        [accuracy\_wine, 1/(1 + mse\_cal)],

        color=['skyblue','salmon'])

plt.title('So sánh hiệu năng giữa mô hình phân lớp & hồi quy')

plt.ylabel('Chỉ số hiệu năng (Càng cao càng tốt)')

plt.grid(True)

plt.show()

# ============================ CẢI THIỆN MÔ HÌNH ============================

# --- CẢI THIỆN WINE - RandomForest ---

rfc = RandomForestClassifier(n\_estimators=100, random\_state=42)

rfc.fit(x\_train\_wine, y\_train\_wine)

y\_pred\_rf = rfc.predict(x\_test\_wine)

acc\_rf = accuracy\_score(y\_test\_wine, y\_pred\_rf)

print(f"Accuracy sau cải thiện (RandomForestClassifier): {acc\_rf:.4f}")

# --- CẢI THIỆN CALIFORNIA - GridSearch GradientBoosting ---

param\_grid = {

    'n\_estimators': [50, 100],

    'learning\_rate': [0.05, 0.1],

    'max\_depth': [3, 4]

}

grid\_model = GridSearchCV(GradientBoostingRegressor(), param\_grid, cv=3, scoring='neg\_mean\_squared\_error')

grid\_model.fit(x\_train\_cal, y\_train\_cal)

print(f"Thông số tốt nhất: {grid\_model.best\_params\_}")

y\_pred\_grid = grid\_model.predict(x\_test\_cal)

mse\_grid = mean\_squared\_error(y\_test\_cal, y\_pred\_grid)

print(f"MSE sau cải thiện (GradientBoosting + GridSearch): {mse\_grid:.4f}")

# --- VẼ BIỂU ĐỒ SO SÁNH SAU CẢI THIỆN ---

plt.figure(figsize=(8,5))

model\_names = ['DT (cũ)', 'RF (mới)', 'GBR (cũ)', 'GBR+Grid (mới)']

metrics = [accuracy\_wine, acc\_rf, 1/(1 + mse\_cal), 1/(1 + mse\_grid)]

colors = ['gray', 'green', 'orange', 'blue']

plt.bar(model\_names, metrics, color=colors)

plt.title("So sánh mô hình trước và sau cải thiện")

plt.ylabel("Chỉ số hiệu năng (Accuracy hoặc 1/(1+MSE))")

plt.grid(True)

plt.show()

**Đề 3 :**

**Câu 1: 4.0 điểm**

Sử dụng bộ dữ liệu Breast Cancer trong thư viện Scikit-Learn để thực hiện các bước

sau:

1. Tải bộ dữ liệu Breast Cancer và hiển thị thông tin cơ bản về bộ dữ liệu này.

2. Chia bộ dữ liệu thành tập huấn luyện mô hình và tập đánh giá hiệu năng của mô

hình.

3. Áp dụng mô hình Support Vector Machine (SVM) từ thư viện Scikit-Learn để huấn luyện trên tập dữ liệu huấn luyện. (Gơi ý từ sau khi làm xong dùng sk.svm import SVC nhé )

4. Đánh giá hiệu năng của mô hình đã huấn luyện trên bộ dữ liệu đánh giá hiệu năng

của mô hình, sử dụng độ đo chính xác (Accuracy).

1. Vẽ biểu đồ phân phối dự đoán của mô hình trên tập kiểm tra.

Note: Đại khái thì trừ cái lưu ý ý 3 và ý 5 vẫn là biểu đồ ma trận nhầm lẫn ra thì đại khái câu 1 này làm vẫn như thế á

**Câu 2: 4.0 điểm**

Sử dụng bộ dữ liệu Diabetes trong thư viện sklearn.datasets để thực hiện các bước

sau:

1. Tải bộ dữ liệu Diabetes và hiển thị thông tin cơ bản về bộ dữ liệu này.

2. Chia bộ dữ liệu Diabetes thành tập dữ liệu huấn luyện và tập dữ liệu đánh giá hiệu

năng mô hình.

3. Áp dụng mô hình Random Forest Regression từ thư viện Scikit-Learn để huấn

luyện trên tập huấn luyện.

4. Đánh giá mô hình dựa trên tập dữ liệu đánh giá hiệu năng của mô hình bằng cách

tính toán sai số trung bình (Mean Absolute Error – MAE).

1. Vẽ biểu đồ scatter plot so sánh giá trị dự đoán và giá trị thực tế trên tập kiểm tra

Note : Câu 2 có khác 1 chút là MAE thay vì MSE như lúc trước nhưng đám thư viện vẫn thế còn đánh giá trên Random Forest Resfression thì như thường dùng ensemble và gọi ra thôi

**Câu 3: 2.0 điểm.**

1. So sánh hiệu năng giữa mô hình SVM và mô hình Random Forest Regression sử

dụng các độ đo phù hợp.

2. Đề xuất một phương pháp cải thiện hiệu năng của mô hình phân lớp hoặc mô hình

hồi quy dựa trên các kết quả thực hiện được trong phần 1 và phần 2.

# ==== CÂU 3: SO SÁNH HIỆU NĂNG & CẢI THIỆN MÔ HÌNH ====

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy\_score, mean\_squared\_error

from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

from sklearn.linear\_model import RidgeCV

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.datasets import load\_breast\_cancer, fetch\_california\_housing

# -----------------------------

# PHẦN 1: Huấn luyện lại mô hình Câu 1 - SVM trên Breast Cancer

# -----------------------------

cancer = load\_breast\_cancer()

X\_cancer = cancer.data

y\_cancer = cancer.target

X\_train\_cancer, X\_test\_cancer, y\_train\_cancer, y\_test\_cancer = train\_test\_split(X\_cancer, y\_cancer, test\_size=0.2, random\_state=42)

svm\_model = SVC(kernel='linear')

svm\_model.fit(X\_train\_cancer, y\_train\_cancer)

y\_pred\_svm = svm\_model.predict(X\_test\_cancer)

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

print(f"Độ chính xác (Accuracy) của SVM trên Breast Cancer: {accuracy\_svm:.4f}")

# -----------------------------

# PHẦN 2: Huấn luyện mô hình hồi quy - Random Forest Regressor trên California Housing

# -----------------------------

california = fetch\_california\_housing()

X\_california = california.data

y\_california = california.target

X\_train\_cal, X\_test\_cal, y\_train\_cal, y\_test\_cal = train\_test\_split(X\_california, y\_california, test\_size=0.2, random\_state=42)

rfr\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)

rfr\_model.fit(X\_train\_cal, y\_train\_cal)

y\_pred\_rfr = rfr\_model.predict(X\_test\_cal)

mse\_rfr = mean\_squared\_error(y\_test\_cal, y\_pred\_rfr)

print(f"MSE của Random Forest Regressor trên California Housing: {mse\_rfr:.4f}")

# -----------------------------

# PHẦN 3: Biểu đồ so sánh hiệu năng

# -----------------------------

plt.figure(figsize=(8,5))

plt.bar(['SVM (Accuracy)', 'RandomForestReg (1 / (1 + MSE))'],

        [accuracy\_svm, 1 / (1 + mse\_rfr)], color=['lightcoral', 'lightseagreen'])

plt.title('So sánh hiệu năng: SVM vs RandomForest Regressor')

plt.ylabel('Chỉ số hiệu năng (Càng cao càng tốt)')

plt.grid(True)

plt.show()

# -----------------------------

# PHẦN 4: CẢI THIỆN MÔ HÌNH PHÂN LOẠI - RandomForestClassifier

# -----------------------------

print("\n--- CẢI THIỆN MÔ HÌNH PHÂN LOẠI (Breast Cancer) ---")

rf\_cls = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_cls.fit(X\_train\_cancer, y\_train\_cancer)

y\_pred\_rf\_cls = rf\_cls.predict(X\_test\_cancer)

accuracy\_rf\_cls = accuracy\_score(y\_test\_cancer, y\_pred\_rf\_cls)

print(f"Accuracy sau khi cải thiện (RandomForestClassifier): {accuracy\_rf\_cls:.4f}")

# -----------------------------

# PHẦN 5: CẢI THIỆN MÔ HÌNH HỒI QUY - RidgeCV

# -----------------------------

print("\n--- CẢI THIỆN MÔ HÌNH HỒI QUY (California Housing) ---")

ridge\_model = RidgeCV(alphas=[0.1, 1.0, 10.0])

ridge\_model.fit(X\_train\_cal, y\_train\_cal)

y\_pred\_ridge = ridge\_model.predict(X\_test\_cal)

mse\_ridge = mean\_squared\_error(y\_test\_cal, y\_pred\_ridge)

print(f"MSE sau khi cải thiện (RidgeCV): {mse\_ridge:.4f}")

# -----------------------------

# PHẦN 6: Biểu đồ so sánh hiệu năng trước/sau cải thiện

# -----------------------------

plt.figure(figsize=(8,5))

models = ['SVM', 'RF\_Classifier', 'RF\_Regressor', 'RidgeCV']

scores = [accuracy\_svm, accuracy\_rf\_cls, 1 / (1 + mse\_rfr), 1 / (1 + mse\_ridge)]

colors = ['salmon', 'green', 'orange', 'blue']

plt.bar(models, scores, color=colors)

plt.title("So sánh hiệu năng trước & sau cải thiện")

plt.ylabel("Chỉ số hiệu năng (Accuracy hoặc 1 / (1 + MSE))")

plt.grid(True)

plt.show()

**Đề 4 :**

**Câu 1: 4.0 điểm**

Sử dụng bộ dữ liệu Iris trong thư viện Scikit-Learn để thực hiện các bước sau:

a. Tải bộ dữ liệu Iris và hiển thị thông tin cơ bản về bộ dữ liệu này.

b. Chia bộ dữ liệu thành hai tập gồm: tập dữ liệu huấn luyện và tập dữ liệu đánh giá mô hình.

c. Áp dụng mô hình Logistic Regression từ thư viện Scikit-Learn để huấn luyện trên tập dữ liệu huấn luyện.

d. Đánh giá hiệu năng của mô hình trên tập dữ liệu đánh giá mô hình, sử dụng độ đo độ chính xác (Accuracy).

**Câu 2: 4.0 điểm**

Sử dụng bộ dữ liệu Boston House Prices trong thư viện Scikit-Learn để thực hiện

các bước sau:

1. Tải bộ dữ liệu Boston House Prices và hiển thị thông tin cơ bản về bộ dữ liệu này.

2. Chia bộ dữ liệu Boston House Prices thành tập huấn luyện và tập dữ liệu đánh giá

hiệu năng mô hình.

3. Áp dụng mô hình Linear Regression từ thư viện Scikit-Learn để huấn luyện trên

bộ dữ liệu huấn luyện.

4. Đánh giá mô hình dựa trên tập dữ liệu đánh giá hiệu năng của mô hình bằng cách

tính toán sai số trung bình (Mean Absolute Error – MAE).

5. Vẽ biểu đồ scatter plot so sánh giá trị dự đoán và giá trị thực tế trên bộ dữ liệu

đánh giá mô hình.

Note : Tạm thời câu 1 - 2 vẫn kiểu kiểu như thế khác ở Câu 3 thôi

**Câu 3: 2.0 điểm.**

1. So sánh hiệu năng giữa mô hình Logistic Regression và mô hình Linear

Regression sử dụng các độ đo hiệu năng phù hợp.

2. Đề xuất một phương pháp cải tiến hiệu năng của mô hình phân lớp hoặc mô hình

hồi quy dựa trên các kết quả thực hiện được trong phần 1 và phần 2.

# ==== CÂU 3: SO SÁNH HIỆU NĂNG & CẢI THIỆN MÔ HÌNH ====

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy\_score, mean\_squared\_error, mean\_absolute\_error

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import RidgeCV, LinearRegression

from sklearn.datasets import load\_iris, load\_boston

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

# -----------------------------

# PHẦN 1: Huấn luyện lại mô hình Câu 1 - Logistic Regression (Iris)

# -----------------------------

iris = load\_iris()

X\_iris = iris.data

y\_iris = iris.target

X\_train\_iris, X\_test\_iris, y\_train\_iris, y\_test\_iris = train\_test\_split(X\_iris, y\_iris, test\_size=0.2, random\_state=42)

lr\_model = LogisticRegression(max\_iter=200)

lr\_model.fit(X\_train\_iris, y\_train\_iris)

y\_pred\_lr = lr\_model.predict(X\_test\_iris)

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

print(f"Độ chính xác (Accuracy) của Logistic Regression trên tập Iris: {accuracy\_lr:.4f}")

# -----------------------------

# PHẦN 2: Huấn luyện lại mô hình Câu 2 - Linear Regression (Boston Housing)

# -----------------------------

boston = load\_boston()

X\_boston = boston.data

y\_boston = boston.target

X\_train\_boston, X\_test\_boston, y\_train\_boston, y\_test\_boston = train\_test\_split(X\_boston, y\_boston, test\_size=0.2, random\_state=42)

lin\_model = LinearRegression()

lin\_model.fit(X\_train\_boston, y\_train\_boston)

y\_pred\_lin = lin\_model.predict(X\_test\_boston)

mae\_lin = mean\_absolute\_error(y\_test\_boston, y\_pred\_lin)

print(f"MAE của Linear Regression trên Boston Housing: {mae\_lin:.4f}")

# -----------------------------

# PHẦN 3: Biểu đồ so sánh hiệu năng hai mô hình

# -----------------------------

plt.figure(figsize=(8,5))

plt.bar(['LogisticReg (Accuracy)', 'LinearReg (1 / (1 + MAE))'],

        [accuracy\_lr, 1 / (1 + mae\_lin)], color=['skyblue', 'lightcoral'])

plt.title('So sánh hiệu năng: Logistic Regression vs Linear Regression')

plt.ylabel('Chỉ số hiệu năng (Càng cao càng tốt)')

plt.grid(True)

plt.show()

# -----------------------------

# PHẦN 4: CẢI THIỆN MÔ HÌNH PHÂN LOẠI - RandomForestClassifier

# -----------------------------

print("\n--- CẢI THIỆN MÔ HÌNH PHÂN LOẠI (Iris) ---")

rf\_cls = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_cls.fit(X\_train\_iris, y\_train\_iris)

y\_pred\_rf = rf\_cls.predict(X\_test\_iris)

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

print(f"Accuracy sau khi cải thiện (RandomForestClassifier): {accuracy\_rf:.4f}")

# -----------------------------

# PHẦN 5: CẢI THIỆN MÔ HÌNH HỒI QUY - RidgeCV

# -----------------------------

print("\n--- CẢI THIỆN MÔ HÌNH HỒI QUY (Boston Housing) ---")

ridge\_model = RidgeCV(alphas=[0.1, 1.0, 10.0])

ridge\_model.fit(X\_train\_boston, y\_train\_boston)

y\_pred\_ridge = ridge\_model.predict(X\_test\_boston)

mae\_ridge = mean\_absolute\_error(y\_test\_boston, y\_pred\_ridge)

print(f"MAE sau khi cải thiện (RidgeCV): {mae\_ridge:.4f}")

# -----------------------------

# PHẦN 6: Biểu đồ so sánh hiệu năng trước/sau cải thiện

# -----------------------------

plt.figure(figsize=(8,5))

models = ['LogisticReg', 'RandomForestCls', 'LinearReg', 'RidgeCV']

scores = [accuracy\_lr, accuracy\_rf, 1 / (1 + mae\_lin), 1 / (1 + mae\_ridge)]

colors = ['gray', 'green', 'orange', 'blue']

plt.bar(models, scores, color=colors)

plt.title("So sánh hiệu năng trước & sau cải thiện")

plt.ylabel("Chỉ số hiệu năng (Accuracy hoặc 1 / (1 + MAE))")

plt.grid(True)

plt.show()