**Linear**

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

# 1️⃣ Import thư viện

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import fetch\_california\_housing

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

from sklearn.linear\_model import LinearRegression

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

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

# 2️⃣ Load dữ liệu giá nhà

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

housing = fetch\_california\_housing()

X = pd.DataFrame(housing.data, columns=housing.feature\_names)

y = pd.Series(housing.target, name='MedHouseVal') # giá nhà trung bình

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

# 3️⃣ Chia dữ liệu train/test

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=1

)

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

# 4️⃣ Huấn luyện mô hình Linear Regression

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

model = LinearRegression()

model.fit(X\_train, y\_train)

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

# 5️⃣ Kiểm thử mô hình

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

y\_pred = model.predict(X\_test)

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

# 6️⃣ Hàm tính MAE, MSE

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

def regression\_eval(y\_test, y\_pred):

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

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

return mae, mse

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

# 7️⃣ Đánh giá mô hình

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

errors = regression\_eval(y\_test, y\_pred)

r2\_total = model.score(X, y)

r2\_train = model.score(X\_train, y\_train)

r2\_test = model.score(X\_test, y\_test)

print(f'MAE = {errors[0]:.3f}')

print(f'MSE = {errors[1]:.3f}')

print(f'R2\_Total giải thích được {(r2\_total \* 100):.1f}% toàn bộ dữ liệu')

print(f'R2\_Train giải thích được {(r2\_train \* 100):.1f}% training set')

print(f'R2\_Test giải thích được {(r2\_test \* 100):.1f}% test set')

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

# 8️⃣ Biểu diễn trực quan

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

x\_min, x\_max = min(y\_pred), max(y\_pred)

y\_min, y\_max = min(y\_test), max(y\_test)

lower, upper = min(x\_min, y\_min), max(x\_max, y\_max)

plt.scatter(y\_pred, y\_test, alpha=0.5)

plt.xlabel('Giá trị dự đoán (Prediction)')

plt.ylabel('Giá trị thực tế (True value)')

plt.plot([lower, upper], [lower, upper], color='red', linestyle='--') # đường phân giác

plt.title("Prediction vs True value (Housing)")

plt.show()

**Logistic**

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

# 1️⃣ Import thư viện

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_diabetes

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, auc

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

# 2️⃣ Load dữ liệu và chuyển target sang 0/1

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

diabetes = load\_diabetes()

X = pd.DataFrame(diabetes.data, columns=diabetes.feature\_names)

# Chia target theo median: 1 nếu > median, 0 nếu <= median

y = pd.Series((diabetes.target > np.median(diabetes.target)).astype(int), name='target')

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

# 3️⃣ Chia dữ liệu train/test

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=1

)

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

# 4️⃣ Huấn luyện mô hình Logistic Regression

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

model = LogisticRegression(max\_iter=1000)

model.fit(X\_train, y\_train)

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

# 5️⃣ Dự đoán

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

y\_pred = model.predict(X\_test)

y\_proba = model.predict\_proba(X\_test)[:, 1] # xác suất lớp 1

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

# 6️⃣ Đánh giá mô hình

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

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

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

report = classification\_report(y\_test, y\_pred)

print(f"Accuracy = {accuracy:.3f}")

print("Confusion Matrix:")

print(cm)

print("\nClassification Report:")

print(report)

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

# 7️⃣ Biểu diễn ROC Curve

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

fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

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

plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc='lower right')

plt.show()

**KNN**

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

# 1️⃣ Import thư viện

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_diabetes

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, auc

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

# 2️⃣ Load dữ liệu và chuyển target sang nhãn 0/1

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

diabetes = load\_diabetes()

X = pd.DataFrame(diabetes.data, columns=diabetes.feature\_names)

# Chia target theo median: 1 nếu > median, 0 nếu <= median

y = pd.Series((diabetes.target > np.median(diabetes.target)).astype(int), name='target')

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

# 3️⃣ Chia dữ liệu train/test

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=1

)

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

# 4️⃣ Chuẩn hóa dữ liệu (quan trọng cho kNN)

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

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

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

# 5️⃣ Huấn luyện mô hình kNN

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

k = 5 # số lượng láng giềng

model = KNeighborsClassifier(n\_neighbors=k)

model.fit(X\_train\_scaled, y\_train)

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

# 6️⃣ Dự đoán

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

y\_pred = model.predict(X\_test\_scaled)

y\_proba = model.predict\_proba(X\_test\_scaled)[:, 1] # xác suất lớp 1

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

# 7️⃣ Đánh giá mô hình

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

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

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

report = classification\_report(y\_test, y\_pred)

print(f"Accuracy = {accuracy:.3f}")

print("Confusion Matrix:")

print(cm)

print("\nClassification Report:")

print(report)

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

# 8️⃣ Vẽ ROC Curve

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

fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

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

plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve - kNN on Diabetes')

plt.legend(loc='lower right')

plt.show()

**NaiveBay**

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

# 1️⃣ Import thư viện

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_diabetes

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

from sklearn.preprocessing import StandardScaler

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, auc

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

# 2️⃣ Load dữ liệu và chuyển target sang nhãn 0/1

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

diabetes = load\_diabetes()

X = pd.DataFrame(diabetes.data, columns=diabetes.feature\_names)

# Chia target theo median: 1 nếu > median, 0 nếu <= median

y = pd.Series((diabetes.target > np.median(diabetes.target)).astype(int), name='target')

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

# 3️⃣ Chia dữ liệu train/test

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=1

)

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

# 4️⃣ Chuẩn hóa dữ liệu (tốt cho Naive Bayes với Gaussian)

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

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

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

# 5️⃣ Huấn luyện mô hình Naive Bayes

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

model = GaussianNB()

model.fit(X\_train\_scaled, y\_train)

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

# 6️⃣ Dự đoán

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

y\_pred = model.predict(X\_test\_scaled)

y\_proba = model.predict\_proba(X\_test\_scaled)[:, 1] # xác suất lớp 1

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

# 7️⃣ Đánh giá mô hình

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

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

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

report = classification\_report(y\_test, y\_pred)

print(f"Accuracy = {accuracy:.3f}")

print("Confusion Matrix:")

print(cm)

print("\nClassification Report:")

print(report)

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

# 8️⃣ Vẽ ROC Curve

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

fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

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

plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve - Gaussian Naive Bayes on Diabetes')

plt.legend(loc='lower right')

plt.show()

**Decistion tree**

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

# 1️⃣ Import thư viện

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_diabetes

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

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, auc

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

# 2️⃣ Load dữ liệu và chuyển target sang nhãn 0/1

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

diabetes = load\_diabetes()

X = pd.DataFrame(diabetes.data, columns=diabetes.feature\_names)

# Nhị phân: 1 nếu target > median, 0 nếu <= median

y = pd.Series((diabetes.target > np.median(diabetes.target)).astype(int), name='target')

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

# 3️⃣ Chia dữ liệu train/test

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=1

)

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

# 4️⃣ Huấn luyện Decision Tree

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

model = DecisionTreeClassifier(max\_depth=5, random\_state=1) # max\_depth giới hạn để tránh overfitting

model.fit(X\_train, y\_train)

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

# 5️⃣ Dự đoán

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

y\_pred = model.predict(X\_test)

y\_proba = model.predict\_proba(X\_test)[:, 1] # xác suất lớp 1

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

# 6️⃣ Đánh giá mô hình

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

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

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

report = classification\_report(y\_test, y\_pred)

print(f"Accuracy = {accuracy:.3f}")

print("Confusion Matrix:")

print(cm)

print("\nClassification Report:")

print(report)

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

# 7️⃣ Vẽ ROC Curve

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

fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

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

plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve - Decision Tree on Diabetes')

plt.legend(loc='lower right')

plt.show()

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

# 8️⃣ Vẽ cây quyết định trực quan (tuỳ chọn)

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

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

plot\_tree(model, feature\_names=X.columns, class\_names=['0','1'], filled=True, rounded=True)

plt.show()

**KNN linear**

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

# 1️⃣ Import thư viện

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import fetch\_california\_housing

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

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

# 2️⃣ Load dữ liệu giá nhà

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

housing = fetch\_california\_housing()

X = pd.DataFrame(housing.data, columns=housing.feature\_names)

y = pd.Series(housing.target, name='MedHouseVal') # giá nhà trung bình

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

# 3️⃣ Chia dữ liệu train/test

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=1

)

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

# 4️⃣ Chuẩn hóa dữ liệu (quan trọng cho kNN)

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

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

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

# 5️⃣ Huấn luyện kNN Regression

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

k = 5 # số lượng láng giềng

model = KNeighborsRegressor(n\_neighbors=k)

model.fit(X\_train\_scaled, y\_train)

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

# 6️⃣ Dự đoán

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

y\_pred = model.predict(X\_test\_scaled)

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

# 7️⃣ Hàm tính MAE, MSE

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

def regression\_eval(y\_test, y\_pred):

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

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

return mae, mse

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

# 8️⃣ Đánh giá mô hình

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

errors = regression\_eval(y\_test, y\_pred)

r2\_total = r2\_score(y, model.predict(scaler.transform(X))) # R2 toàn bộ dữ liệu

r2\_train = r2\_score(y\_train, model.predict(X\_train\_scaled))

r2\_test = r2\_score(y\_test, y\_pred)

print(f'MAE = {errors[0]:.3f}')

print(f'MSE = {errors[1]:.3f}')

print(f'R2\_Total giải thích được {(r2\_total\*100):.1f}% toàn bộ dữ liệu')

print(f'R2\_Train giải thích được {(r2\_train\*100):.1f}% training set')

print(f'R2\_Test giải thích được {(r2\_test\*100):.1f}% test set')

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

# 9️⃣ Biểu diễn trực quan

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

x\_min, x\_max = min(y\_pred), max(y\_pred)

y\_min, y\_max = min(y\_test), max(y\_test)

lower, upper = min(x\_min, y\_min), max(x\_max, y\_max)

plt.scatter(y\_pred, y\_test, alpha=0.5)

plt.xlabel('Giá trị dự đoán (Prediction)')

plt.ylabel('Giá trị thực tế (True value)')

plt.plot([lower, upper], [lower, upper], color='red', linestyle='--') # đường phân giác

plt.title("Prediction vs True value - kNN Regression on Housing")

plt.show()

**SVM**

# 1. Import thư viện

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# 2. Load dữ liệu

data = pd.read\_csv('diabetes.csv') # đường dẫn tới file CSV

X = data.drop('Outcome', axis=1)

y = data['Outcome']

# 3. Chia dữ liệu train/test

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

# 4. Chuẩn hóa dữ liệu

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# 5. Huấn luyện SVM

svm\_model = SVC(kernel='rbf', C=1.0, gamma='scale') # kernel='linear' nếu muốn thử tuyến tính

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

# 6. Dự đoán

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

# 7. Đánh giá mô hình

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))