# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

Copyright 2025 | Nghiên Cứu Thuốc | www.nghiencuuthuoc.com | Zalo: +84888999311 |

## 🧩 Topic: Dự đoán cách thức gắn ligand bằng docking

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit theo chuẩn MLOps. Dựa trên thông tin bạn cung cấp, chúng ta sẽ cùng nhau xây dựng quy trình làm việc hiệu quả và giải quyết các vấn đề phát sinh.

**1. Phân tích mô hình phân tích (Analysis of the Analysis Model)**

Chúng ta sẽ tập trung vào việc xây dựng một quy trình làm việc rõ ràng, dễ theo dõi và có khả năng mở rộng. Dưới đây là các bước chính:

* **Truy vấn dữ liệu từ ChEMBL 35:** Sử dụng SQL để trích xuất thông tin cần thiết từ cơ sở dữ liệu ChEMBL 35. Chúng ta sẽ tập trung vào các bảng như activities, molecule\_dictionary, và compound\_structures để thu thập dữ liệu về hoạt tính sinh học, thông tin phân tử, và cấu trúc hóa học.
* **Tiền xử lý dữ liệu:** Làm sạch và chuẩn hóa dữ liệu, xử lý các giá trị thiếu, và chuyển đổi dữ liệu về định dạng phù hợp cho phân tích.
* **Tính toán đặc trưng phân tử:** Sử dụng RDKit để tính toán các đặc trưng phân tử (molecular features) từ cấu trúc hóa học của các hợp chất. Các đặc trưng này có thể bao gồm các thuộc tính vật lý hóa học, descriptor cấu trúc, và fingerprint.
* **Phân tích thống kê và mô hình hóa:** Sử dụng các kỹ thuật thống kê và học máy để khám phá mối quan hệ giữa các đặc trưng phân tử và hoạt tính sinh học. Chúng ta có thể sử dụng các mô hình như hồi quy tuyến tính, random forest, hoặc mạng neural để dự đoán hoạt tính của các hợp chất mới.
* **Đánh giá mô hình:** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp như RMSE, R-squared, và AUC.
* **Trực quan hóa dữ liệu:** Sử dụng các công cụ trực quan hóa để khám phá dữ liệu và trình bày kết quả một cách dễ hiểu.

**2. Hướng dẫn song ngữ (Bilingual Guidance)**

**2.1. SQL**

* **Mục tiêu:** Trích xuất dữ liệu hoạt tính sinh học (bioactivity data) và thông tin phân tử (molecule information) từ cơ sở dữ liệu ChEMBL.
* **Ví dụ:**

-- English  
-- Extracting activity data and molecule information for a specific target  
SELECT  
 act.activity\_id,  
 mol.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 cs.canonical\_smiles  
FROM  
 activities act  
JOIN  
 molecule\_dictionary mol ON act.molregno = mol.molregno  
JOIN  
 compound\_structures cs ON mol.molregno = cs.molregno  
WHERE  
 act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Ensure standard\_value contains only numbers and dots  
LIMIT 100;  
  
-- Vietnamese  
-- Trích xuất dữ liệu hoạt tính và thông tin phân tử cho một mục tiêu cụ thể  
SELECT  
 act.activity\_id,  
 mol.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 cs.canonical\_smiles  
FROM  
 activities act  
JOIN  
 molecule\_dictionary mol ON act.molregno = mol.molregno  
JOIN  
 compound\_structures cs ON mol.molregno = cs.molregno  
WHERE  
 act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Đảm bảo standard\_value chỉ chứa số và dấu chấm  
LIMIT 100;

* **Giải thích:**
  + Câu truy vấn này chọn các cột activity\_id, chembl\_id, standard\_type, standard\_value, standard\_units, và canonical\_smiles từ các bảng activities, molecule\_dictionary, và compound\_structures.
  + Nó lọc dữ liệu để chỉ bao gồm các hoạt động có standard\_type là ‘IC50’ và standard\_units là ‘nM’, và standard\_value không rỗng, lớn hơn 0, và chỉ chứa số và dấu chấm.
  + LIMIT 100 giới hạn kết quả trả về 100 dòng.
* **Sửa lỗi:**
  + Lỗi ERROR: operator does not exist: numeric ~ unknown xảy ra do bạn đang cố gắng sử dụng toán tử ~ (regex match) trên cột kiểu số (standard\_value). Để khắc phục, bạn có thể ép kiểu standard\_value sang kiểu text trước khi so sánh:

AND CAST(act.standard\_value AS TEXT) ~ '^[0-9\.]+$'

**2.2. Python**

* **Mục tiêu:** Đọc dữ liệu từ file CSV, tính toán đặc trưng phân tử bằng RDKit, và xây dựng mô hình học máy.
* **Ví dụ:**

# English  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
import os  
import numpy as np  
  
# Define base path  
base\_path = "../data" # Adjust if needed  
  
# Load data from CSV  
csv\_file = "chembl\_activity\_data.csv" # Replace with your actual CSV file name  
data\_path = os.path.join(base\_path, csv\_file)  
df = pd.read\_csv(data\_path)  
  
# Function to calculate molecular weight  
def calculate\_mw(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Descriptors.MolWt(mol)  
 else:  
 return np.nan  
  
# Apply the function to create a new column 'molecular\_weight'  
df['molecular\_weight'] = df['canonical\_smiles'].apply(calculate\_mw)  
  
# Drop rows with NaN values in 'standard\_value' or 'molecular\_weight'  
df = df.dropna(subset=['standard\_value', 'molecular\_weight'])  
  
# Prepare data for modeling  
X = df[['molecular\_weight']] # Feature: Molecular Weight  
y = df['standard\_value'] # Target: standard\_value  
  
# Split data into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Train a linear regression model  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Make predictions on the test set  
y\_pred = model.predict(X\_test)  
  
# Evaluate the model  
mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")  
  
# Vietnamese  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
import os  
import numpy as np  
  
# Định nghĩa đường dẫn gốc  
base\_path = "../data" # Điều chỉnh nếu cần  
  
# Tải dữ liệu từ file CSV  
csv\_file = "chembl\_activity\_data.csv" # Thay thế bằng tên file CSV của bạn  
data\_path = os.path.join(base\_path, csv\_file)  
df = pd.read\_csv(data\_path)  
  
# Hàm tính toán khối lượng phân tử  
def calculate\_mw(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Descriptors.MolWt(mol)  
 else:  
 return np.nan  
  
# Áp dụng hàm để tạo cột mới 'molecular\_weight'  
df['molecular\_weight'] = df['canonical\_smiles'].apply(calculate\_mw)  
  
# Loại bỏ các hàng có giá trị NaN trong 'standard\_value' hoặc 'molecular\_weight'  
df = df.dropna(subset=['standard\_value', 'molecular\_weight'])  
  
# Chuẩn bị dữ liệu cho mô hình hóa  
X = df[['molecular\_weight']] # Đặc trưng: Khối lượng phân tử  
y = df['standard\_value'] # Mục tiêu: standard\_value  
  
# Chia dữ liệu thành tập huấn luyện và tập kiểm tra  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Huấn luyện mô hình hồi quy tuyến tính  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Dự đoán trên tập kiểm tra  
y\_pred = model.predict(X\_test)  
  
# Đánh giá mô hình  
mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")

* **Giải thích:**
  + Đoạn code này đọc dữ liệu từ file CSV, tính toán khối lượng phân tử từ SMILES sử dụng RDKit, và xây dựng một mô hình hồi quy tuyến tính để dự đoán standard\_value dựa trên khối lượng phân tử.
  + Nó chia dữ liệu thành tập huấn luyện và tập kiểm tra, huấn luyện mô hình trên tập huấn luyện, và đánh giá hiệu suất của mô hình trên tập kiểm tra.
* **Sửa lỗi:**
  + Nếu bạn gặp lỗi TypeError: mean\_squared\_error() got an unexpected keyword argument 'squared', điều này có nghĩa là phiên bản scikit-learn của bạn quá cũ. Bạn có thể nâng cấp scikit-learn bằng lệnh pip install -U scikit-learn. Nếu bạn không muốn nâng cấp, bạn có thể bỏ tham số squared=False (trong phiên bản cũ, mặc định squared=True, trả về MSE thay vì RMSE).

**3. Ví dụ code SQL và Python mẫu (Example SQL and Python Code Snippets)**

Dưới đây là 5 ví dụ code SQL và Python mẫu để bạn tham khảo:

**Ví dụ 1: Lọc dữ liệu theo khoảng giá trị (Filtering data by value range)**

* **SQL:**

-- English  
-- Select compounds with IC50 values between 100 and 1000 nM  
SELECT mol.chembl\_id, act.standard\_value  
FROM activities act  
JOIN molecule\_dictionary mol ON act.molregno = mol.molregno  
WHERE act.standard\_type = 'IC50'  
AND act.standard\_units = 'nM'  
AND act.standard\_value >= 100  
AND act.standard\_value <= 1000  
LIMIT 100;  
  
-- Vietnamese  
-- Chọn các hợp chất có giá trị IC50 nằm giữa 100 và 1000 nM  
SELECT mol.chembl\_id, act.standard\_value  
FROM activities act  
JOIN molecule\_dictionary mol ON act.molregno = mol.molregno  
WHERE act.standard\_type = 'IC50'  
AND act.standard\_units = 'nM'  
AND act.standard\_value >= 100  
AND act.standard\_value <= 1000  
LIMIT 100;

* **Python:**

# English  
# Filter DataFrame for IC50 values between 100 and 1000  
filtered\_df = df[(df['standard\_type'] == 'IC50') & (df['standard\_units'] == 'nM') & (df['standard\_value'] >= 100) & (df['standard\_value'] <= 1000)]  
  
# Vietnamese  
# Lọc DataFrame cho các giá trị IC50 nằm giữa 100 và 1000  
filtered\_df = df[(df['standard\_type'] == 'IC50') & (df['standard\_units'] == 'nM') & (df['standard\_value'] >= 100) & (df['standard\_value'] <= 1000)]

**Ví dụ 2: Tính toán logP sử dụng RDKit (Calculating logP using RDKit)**

* **Python:**

# English  
from rdkit.Chem import Crippen  
  
def calculate\_logp(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Crippen.MolLogP(mol)  
 else:  
 return None  
  
df['logp'] = df['canonical\_smiles'].apply(calculate\_logp)  
  
# Vietnamese  
from rdkit.Chem import Crippen  
  
def calculate\_logp(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Crippen.MolLogP(mol)  
 else:  
 return None  
  
df['logp'] = df['canonical\_smiles'].apply(calculate\_logp)

**Ví dụ 3: Trích xuất dữ liệu cho một protein cụ thể (Extracting data for a specific protein)**

* **SQL:**

-- English  
-- Select activity data for a specific target (e.g., with target\_chembl\_id 'CHEMBL205')  
SELECT act.activity\_id, mol.chembl\_id, act.standard\_value  
FROM activities act  
JOIN molecule\_dictionary mol ON act.molregno = mol.molregno  
JOIN target\_dictionary td ON act.tid = td.tid  
WHERE td.target\_chembl\_id = 'CHEMBL205'  
AND act.standard\_type = 'IC50'  
AND act.standard\_units = 'nM'  
LIMIT 100;  
  
-- Vietnamese  
-- Chọn dữ liệu hoạt tính cho một mục tiêu cụ thể (ví dụ, với target\_chembl\_id 'CHEMBL205')  
SELECT act.activity\_id, mol.chembl\_id, act.standard\_value  
FROM activities act  
JOIN molecule\_dictionary mol ON act.molregno = mol.molregno  
JOIN target\_dictionary td ON act.tid = td.tid  
WHERE td.target\_chembl\_id = 'CHEMBL205'  
AND act.standard\_type = 'IC50'  
AND act.standard\_units = 'nM'  
LIMIT 100;

**Ví dụ 4: Sử dụng Random Forest để dự đoán hoạt tính (Using Random Forest to predict activity)**

* **Python:**

# English  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Assuming you have features X and target y  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
y\_pred = model.predict(X\_test)  
  
mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")  
  
# Vietnamese  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Giả sử bạn có các đặc trưng X và mục tiêu y  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
y\_pred = model.predict(X\_test)  
  
mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")

**Ví dụ 5: Tính toán Tanimoto similarity giữa các phân tử (Calculating Tanimoto similarity between molecules)**

* **Python:**

# English  
from rdkit import DataStructs  
from rdkit.Chem import AllChem  
  
def calculate\_tanimoto\_similarity(smiles1, smiles2):  
 mol1 = Chem.MolFromSmiles(smiles1)  
 mol2 = Chem.MolFromSmiles(smiles2)  
  
 if mol1 and mol2:  
 fp1 = AllChem.GetMorganFingerprintAsBitVect(mol1, 2, nBits=2048)  
 fp2 = AllChem.GetMorganFingerprintAsBitVect(mol2, 2, nBits=2048)  
 return DataStructs.TanimotoSimilarity(fp1, fp2)  
 else:  
 return None  
  
# Example usage:  
smiles1 = "CC(=O)Oc1ccccc1C(=O)O" # Aspirin  
smiles2 = "CC(=O)Nc1ccccc1O" # Paracetamol  
similarity = calculate\_tanimoto\_similarity(smiles1, smiles2)  
print(f"Tanimoto Similarity: {similarity}")  
  
# Vietnamese  
from rdkit import DataStructs  
from rdkit.Chem import AllChem  
  
def calculate\_tanimoto\_similarity(smiles1, smiles2):  
 mol1 = Chem.MolFromSmiles(smiles1)  
 mol2 = Chem.MolFromSmiles(smiles2)  
  
 if mol1 and mol2:  
 fp1 = AllChem.GetMorganFingerprintAsBitVect(mol1, 2, nBits=2048)  
 fp2 = AllChem.GetMorganFingerprintAsBitVect(mol2, 2, nBits=2048)  
 return DataStructs.TanimotoSimilarity(fp1, fp2)  
 else:  
 return None  
  
# Ví dụ sử dụng:  
smiles1 = "CC(=O)Oc1ccccc1C(=O)O" # Aspirin  
smiles2 = "CC(=O)Nc1ccccc1O" # Paracetamol  
similarity = calculate\_tanimoto\_similarity(smiles1, smiles2)  
print(f"Tanimoto Similarity: {similarity}")

**Lưu ý quan trọng:**

* Hãy thay thế các tên file và đường dẫn phù hợp với cấu trúc thư mục của bạn.
* Điều chỉnh các tham số của mô hình học máy để đạt được hiệu suất tốt nhất.
* Sử dụng các kỹ thuật tiền xử lý dữ liệu phù hợp để làm sạch và chuẩn hóa dữ liệu.
* Thường xuyên kiểm tra và đánh giá mô hình của bạn để đảm bảo rằng nó hoạt động tốt.
* Khi gặp lỗi, hãy đọc kỹ thông báo lỗi và tìm kiếm giải pháp trên Google hoặc Stack Overflow.

Chúc bạn thành công với dự án của mình! Nếu bạn có bất kỳ câu hỏi nào, đừng ngần ngại hỏi.