# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Ứng dụng tìm kiếm tương đồng tương tác

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 này. Với vai trò là một chuyên gia R&D dược học, tôi sẽ giúp bạn xây dựng quy trình phân tích hiệu quả, khắc phục các lỗi và đưa ra các ví dụ cụ thể để bạn có thể bắt đầu ngay.

**Phân tích Tổng quan và Mô hình Phân tích (Overview and Analysis Model)**

Dự án của bạn tập trung vào việc khai thác dữ liệu từ cơ sở dữ liệu ChEMBL 35, sử dụng RDKit để xử lý thông tin hóa học và PostgreSQL để quản lý dữ liệu. Mục tiêu là hỗ trợ các nghiên cứu và phát triển thuốc. Dưới đây là mô hình phân tích tổng quan mà chúng ta có thể áp dụng:

1. **Data Extraction and Preparation (Trích xuất và Chuẩn bị Dữ liệu):**
   * Sử dụng SQL để truy vấn dữ liệu từ ChEMBL 35, tập trung vào các bảng chứa thông tin về hoạt tính sinh học (bioactivity), cấu trúc hóa học, và các thuộc tính liên quan.
   * Lưu dữ liệu trích xuất vào các file CSV để dễ dàng xử lý bằng Python.
   * Sử dụng RDKit để tính toán các descriptor phân tử (ví dụ: trọng lượng phân tử, logP, số lượng vòng) từ cấu trúc SMILES.
2. **Data Cleaning and Preprocessing (Làm sạch và Tiền xử lý Dữ liệu):**
   * Xử lý các giá trị thiếu (missing values) và ngoại lệ (outliers).
   * Chuẩn hóa dữ liệu để đảm bảo tính nhất quán.
   * Chuyển đổi dữ liệu định tính thành định lượng nếu cần thiết.
3. **Exploratory Data Analysis (EDA) (Phân tích Khám phá Dữ liệu):**
   * Sử dụng các kỹ thuật thống kê và trực quan hóa để khám phá các xu hướng và mối quan hệ trong dữ liệu.
   * Ví dụ: phân phối của các giá trị hoạt tính, mối tương quan giữa các descriptor phân tử và hoạt tính.
4. **Model Building and Evaluation (Xây dựng và Đánh giá Mô hình):**
   * Xây dựng các mô hình dự đoán hoạt tính (ví dụ: hồi quy tuyến tính, random forest, mạng nơ-ron) dựa trên các descriptor phân tử.
   * Đánh giá hiệu suất của mô hình bằng các chỉ số phù hợp (ví dụ: R-squared, RMSE, AUC).
5. **Interpretation and Application (Giải thích và Ứng dụng):**
   * Giải thích các kết quả mô hình để hiểu rõ hơn về mối quan hệ giữa cấu trúc và hoạt tính.
   * Ứng dụng mô hình để dự đoán hoạt tính của các hợp chất mới, sàng lọc ảo (virtual screening), và tối ưu hóa cấu trúc thuốc.

**Hướng dẫn Song ngữ và Mã (Bilingual Guide and Code)**

Dưới đây là hướng dẫn chi tiết và mã mẫu cho từng bước, kèm theo giải thích bằng cả tiếng Anh và tiếng Việt:

**1. Data Extraction (Trích xuất Dữ liệu)**

**SQL Code (Tiếng Anh):**

-- Extract bioactivity data and molecule structures from ChEMBL  
-- Lấy dữ liệu hoạt tính sinh học và cấu trúc phân tử từ ChEMBL  
SELECT  
 act.molregno,  
 mol.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 mol.molsmiles  
FROM  
 activities act  
JOIN  
 molecule\_dictionary mol ON act.molregno = mol.molregno  
WHERE  
 act.standard\_type = 'IC50' -- You can change this to other activity types like Ki, EC50, etc.  
 AND act.standard\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0  
 AND act.standard\_value < 10000 -- Filter out very high IC50 values  
 AND act.standard\_units = 'nM'  
LIMIT 100; -- Limit to 100 rows for testing  
  
-- Save the result as a CSV file (e.g., chembl\_ic50\_data.csv)  
-- Lưu kết quả vào file CSV (ví dụ: chembl\_ic50\_data.csv)

**SQL Code (Tiếng Việt - Giải thích):**

-- Lấy dữ liệu hoạt tính sinh học và cấu trúc phân tử từ ChEMBL  
SELECT  
 act.molregno, -- Mã số phân tử (molecule registry number)  
 mol.chembl\_id, -- Mã ChEMBL của phân tử  
 act.standard\_type, -- Loại hoạt tính (ví dụ: IC50, Ki)  
 act.standard\_value, -- Giá trị hoạt tính  
 act.standard\_units, -- Đơn vị của giá trị hoạt tính  
 mol.molsmiles -- Cấu trúc SMILES của phân tử  
FROM  
 activities act -- Bảng chứa thông tin hoạt tính  
JOIN  
 molecule\_dictionary mol ON act.molregno = mol.molregno -- Bảng chứa thông tin phân tử, kết nối qua molregno  
WHERE  
 act.standard\_type = 'IC50' -- Lọc chỉ lấy các hoạt tính loại IC50 (có thể thay đổi)  
 AND act.standard\_relation = '=' -- Lọc chỉ lấy các hoạt tính có quan hệ "=" (bằng)  
 AND act.standard\_value IS NOT NULL -- Lọc bỏ các giá trị hoạt tính bị thiếu  
 AND act.standard\_value > 0 -- Lọc bỏ các giá trị hoạt tính âm hoặc bằng 0  
 AND act.standard\_value < 10000 -- Lọc bỏ các giá trị IC50 quá cao (lớn hơn 10000 nM)  
 AND act.standard\_units = 'nM' -- Lọc chỉ lấy các hoạt tính có đơn vị là nM (nanomolar)  
LIMIT 100; -- Giới hạn số lượng kết quả trả về là 100 dòng (cho mục đích thử nghiệm)  
  
-- Lưu kết quả vào file CSV (ví dụ: chembl\_ic50\_data.csv)

**Lưu ý về lỗi “operator does not exist: numeric ~ unknown”:**

Lỗi này thường xảy ra khi bạn cố gắng sử dụng toán tử ~ (regular expression match) trên một cột kiểu số (numeric). Để khắc phục, bạn có thể chuyển đổi cột số thành kiểu chuỗi (text) trước khi sử dụng toán tử ~. Tuy nhiên, trong trường hợp này, bạn không cần sử dụng regular expression để lọc các giá trị số. Bạn có thể sử dụng các toán tử so sánh trực tiếp như > và <.

**2. Data Loading and Processing with RDKit (Tải và Xử lý Dữ liệu với RDKit)**

**Python Code (Tiếng Anh):**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import os  
  
# Define the base path for your project  
# Định nghĩa đường dẫn gốc của dự án  
base\_path = '.' # Adjust this to your actual base path  
  
# Load the CSV file containing ChEMBL data  
# Tải file CSV chứa dữ liệu ChEMBL  
data\_path = os.path.join(base\_path, 'data', 'chembl\_ic50\_data.csv') # Adjust the file name accordingly  
df = pd.read\_csv(data\_path)  
  
# Function to calculate molecular descriptors using RDKit  
# Hàm tính toán các descriptor phân tử bằng RDKit  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 descriptors = {  
 "MW": Descriptors.MolWt(mol),  
 "LogP": Descriptors.MolLogP(mol),  
 "HBA": Descriptors.NumHAcceptors(mol),  
 "HBD": Descriptors.NumHDonors(mol),  
 "TPSA": Descriptors.TPSA(mol)  
 }  
 return descriptors  
  
# Apply the function to each SMILES string in the dataframe  
# Áp dụng hàm cho mỗi chuỗi SMILES trong dataframe  
df['descriptors'] = df['molsmiles'].apply(calculate\_descriptors)  
  
# Expand the descriptor dictionary into separate columns  
# Mở rộng từ điển descriptor thành các cột riêng biệt  
df = pd.concat([df, df['descriptors'].apply(pd.Series)], axis=1)  
  
# Drop rows with missing descriptors  
# Loại bỏ các hàng có descriptor bị thiếu  
df = df.dropna(subset=['MW', 'LogP', 'HBA', 'HBD', 'TPSA'])  
  
# Convert IC50 values to pIC50 (optional)  
# Chuyển đổi giá trị IC50 sang pIC50 (tùy chọn)  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Convert nM to Molar  
  
# Display the first few rows of the processed dataframe  
# Hiển thị một vài hàng đầu tiên của dataframe đã xử lý  
print(df.head())

**Python Code (Tiếng Việt - Giải thích):**

import pandas as pd # Thư viện để làm việc với dữ liệu dạng bảng  
from rdkit import Chem # Thư viện RDKit để xử lý thông tin hóa học  
from rdkit.Chem import Descriptors # Các module con của RDKit để tính toán descriptor  
import os # Thư viện để tương tác với hệ điều hành  
  
# Định nghĩa đường dẫn gốc của dự án  
base\_path = '.' # Điều chỉnh đường dẫn này cho phù hợp với cấu trúc thư mục của bạn  
  
# Tải file CSV chứa dữ liệu ChEMBL  
data\_path = os.path.join(base\_path, 'data', 'chembl\_ic50\_data.csv') # Điều chỉnh tên file cho phù hợp  
df = pd.read\_csv(data\_path) # Đọc dữ liệu từ file CSV vào một dataframe  
  
# Hàm tính toán các descriptor phân tử bằng RDKit  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles) # Chuyển đổi chuỗi SMILES thành đối tượng phân tử của RDKit  
 if mol is None: # Kiểm tra xem chuyển đổi có thành công không  
 return None # Trả về None nếu không thành công (ví dụ: SMILES không hợp lệ)  
 descriptors = { # Tạo một từ điển để lưu trữ các descriptor  
 "MW": Descriptors.MolWt(mol), # Trọng lượng phân tử (Molecular Weight)  
 "LogP": Descriptors.MolLogP(mol), # Hệ số phân vùng octanol-nước (LogP)  
 "HBA": Descriptors.NumHAcceptors(mol), # Số lượng chất nhận liên kết hydro (Hydrogen Bond Acceptors)  
 "HBD": Descriptors.NumHDonors(mol), # Số lượng chất cho liên kết hydro (Hydrogen Bond Donors)  
 "TPSA": Descriptors.TPSA(mol) # Diện tích bề mặt phân cực (Topological Polar Surface Area)  
 }  
 return descriptors # Trả về từ điển chứa các descriptor  
  
# Áp dụng hàm cho mỗi chuỗi SMILES trong dataframe  
df['descriptors'] = df['molsmiles'].apply(calculate\_descriptors) # Tạo một cột mới 'descriptors' chứa kết quả  
  
# Mở rộng từ điển descriptor thành các cột riêng biệt  
df = pd.concat([df, df['descriptors'].apply(pd.Series)], axis=1) # Tạo các cột mới từ từ điển descriptor  
  
# Loại bỏ các hàng có descriptor bị thiếu  
df = df.dropna(subset=['MW', 'LogP', 'HBA', 'HBD', 'TPSA']) # Loại bỏ các hàng có giá trị NaN trong các cột descriptor  
  
# Chuyển đổi giá trị IC50 sang pIC50 (tùy chọn)  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Chuyển đổi nM sang Molar và tính pIC50  
  
# Hiển thị một vài hàng đầu tiên của dataframe đã xử lý  
print(df.head()) # In ra 5 hàng đầu tiên của dataframe

**3. Model Building (Xây dựng Mô hình)**

**Python Code (Tiếng Anh):**

from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Prepare the data for modeling  
# Chuẩn bị dữ liệu cho việc xây dựng mô hình  
X = df[['MW', 'LogP', 'HBA', 'HBD', 'TPSA']] # Features  
y = df['pIC50'] # Target variable  
  
# Split the data into training and testing sets  
# 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)  
  
# Create a Random Forest Regressor model  
# Tạo một mô hình Random Forest Regressor  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
  
# Train the model  
# Huấn luyện mô hình  
model.fit(X\_train, y\_train)  
  
# Make predictions on the test set  
# Dự đoán trên tập kiểm tra  
y\_pred = model.predict(X\_test)  
  
# Evaluate the model  
# 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}")

**Python Code (Tiếng Việt - Giải thích):**

from sklearn.model\_selection import train\_test\_split # Hàm để chia dữ liệu  
from sklearn.ensemble import RandomForestRegressor # Mô hình Random Forest  
from sklearn.metrics import mean\_squared\_error, r2\_score # Các chỉ số đánh giá mô hình  
  
# Chuẩn bị dữ liệu cho việc xây dựng mô hình  
X = df[['MW', 'LogP', 'HBA', 'HBD', 'TPSA']] # Các biến độc lập (features)  
y = df['pIC50'] # Biến phụ thuộc (target variable)  
  
# 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) # 80% cho huấn luyện, 20% cho kiểm tra  
  
# Tạo một mô hình Random Forest Regressor  
model = RandomForestRegressor(n\_estimators=100, random\_state=42) # Số lượng cây trong rừng là 100  
  
# Huấn luyện mô hình  
model.fit(X\_train, y\_train) # Sử dụng tập huấn luyện để "dạy" mô hình  
  
# Dự đoán trên tập kiểm tra  
y\_pred = model.predict(X\_test) # Sử dụng mô hình đã huấn luyện để dự đoán giá trị pIC50 trên tập kiểm tra  
  
# Đánh giá mô hình  
mse = mean\_squared\_error(y\_test, y\_pred) # Tính sai số bình phương trung bình (Mean Squared Error)  
r2 = r2\_score(y\_test, y\_pred) # Tính hệ số xác định (R-squared)  
  
print(f"Mean Squared Error: {mse}") # In ra giá trị MSE  
print(f"R-squared: {r2}") # In ra giá trị R-squared

**Lưu ý về lỗi “squared=False trong hàm mean\_squared\_error”:**

Tham số squared=False đã được thêm vào hàm mean\_squared\_error trong các phiên bản scikit-learn mới hơn để trả về Root Mean Squared Error (RMSE) thay vì MSE. Nếu bạn đang sử dụng phiên bản scikit-learn cũ, bạn sẽ gặp lỗi khi sử dụng tham số này. Để khắc phục, bạn có thể nâng cấp scikit-learn lên phiên bản mới nhất hoặc tính RMSE thủ công bằng cách lấy căn bậc hai của MSE:

mse = mean\_squared\_error(y\_test, y\_pred)  
rmse = mse\*\*0.5  
print(f"Root Mean Squared Error: {rmse}")

**4. Ví dụ Mã (Code Examples)**

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

**Ví dụ 1: Lọc dữ liệu theo khoảng giá trị hoạt tính (Filtering Data by Activity Range)**

**SQL:**

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\_value BETWEEN 100 AND 1000;

**Python:**

df\_filtered = df[(df['standard\_value'] >= 100) & (df['standard\_value'] <= 1000)]  
print(df\_filtered.head())

**Ví dụ 2: Tính toán số lượng phân tử có một số lượng vòng nhất định (Calculating Number of Molecules with Specific Ring Count)**

**SQL (Cần thêm thông tin về số lượng vòng vào bảng):**

-- This requires a table with ring count information  
-- Điều này yêu cầu một bảng có thông tin về số lượng vòng  
-- Example: molecule\_properties table with 'num\_rings' column  
-- Ví dụ: bảng molecule\_properties có cột 'num\_rings'  
SELECT mol.chembl\_id  
FROM molecule\_dictionary mol  
JOIN molecule\_properties mp ON mol.molregno = mp.molregno  
WHERE mp.num\_rings = 3;

**Python:**

from rdkit.Chem import RingInfo  
  
def get\_ring\_count(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 ring\_info = mol.GetRingInfo()  
 return ring\_info.NumRings()  
 return None  
  
df['ring\_count'] = df['molsmiles'].apply(get\_ring\_count)  
print(df['ring\_count'].value\_counts())

**Ví dụ 3: Tìm các phân tử có trọng lượng phân tử trong một khoảng nhất định (Finding Molecules with Molecular Weight in a Specific Range)**

**SQL:**

-- Assuming you have a table with molecular weights  
-- Giả sử bạn có một bảng với trọng lượng phân tử  
-- Example: molecule\_properties table with 'molecular\_weight' column  
-- Ví dụ: bảng molecule\_properties có cột 'molecular\_weight'  
SELECT mol.chembl\_id  
FROM molecule\_dictionary mol  
JOIN molecule\_properties mp ON mol.molregno = mp.molregno  
WHERE mp.molecular\_weight BETWEEN 200 AND 400;

**Python:**

df\_mw\_filtered = df[(df['MW'] >= 200) & (df['MW'] <= 400)]  
print(df\_mw\_filtered.head())

**Ví dụ 4: Phân tích mối tương quan giữa LogP và pIC50 (Analyzing Correlation between LogP and pIC50)**

**SQL (Không thể tính tương quan trực tiếp trong SQL đơn giản):**

-- SQL can't directly calculate correlation easily, use Python  
-- SQL không thể tính tương quan trực tiếp một cách dễ dàng, hãy sử dụng Python  
SELECT mol.chembl\_id, act.standard\_value, mp.logp  
FROM activities act  
JOIN molecule\_dictionary mol ON act.molregno = mol.molregno  
JOIN molecule\_properties mp ON mol.molregno = mp.molregno  
WHERE act.standard\_type = 'IC50';

**Python:**

correlation = df['LogP'].corr(df['pIC50'])  
print(f"Correlation between LogP and pIC50: {correlation}")

**Ví dụ 5: Trực quan hóa dữ liệu sử dụng scatter plot (Data Visualization using Scatter Plot)**

**Python:**

import matplotlib.pyplot as plt  
  
plt.scatter(df['LogP'], df['pIC50'])  
plt.xlabel('LogP')  
plt.ylabel('pIC50')  
plt.title('LogP vs pIC50')  
plt.show()

**Lưu ý Quan trọng:**

* **Điều chỉnh đường dẫn:** Hãy chắc chắn rằng bạn đã điều chỉnh đường dẫn file và tên file trong code cho phù hợp với cấu trúc thư mục và tên file thực tế của bạn.
* **Cài đặt thư viện:** Đảm bảo bạn đã cài đặt tất cả các thư viện cần thiết (ví dụ: pandas, rdkit, scikit-learn, matplotlib) trước khi chạy code Python.
* **Xử lý lỗi:** Luôn kiểm tra và xử lý các lỗi có thể xảy ra trong quá trình chạy code.
* **Giới hạn dữ liệu:** Vì lý do hiệu suất, chúng ta đang giới hạn số lượng dữ liệu được trích xuất và xử lý. Bạn có thể tăng giới hạn này khi cần thiết, nhưng hãy cẩn thận với tài nguyên máy tính của bạn.

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