# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Phân nhóm theo cơ chế tác dụng của thuốc ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Với vai trò là một chuyên gia nghiên cứu và phát triển dược học, tôi sẽ cung cấp cho bạn các phân tích, hướng dẫn song ngữ (Việt - Anh), code SQL và Python mẫu, cùng với các lưu ý để giúp bạn vượt qua các lỗi thường gặp.

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

Dự án của bạn tập trung vào việc khai thác dữ liệu từ ChEMBL 35, một cơ sở dữ liệu lớn chứa thông tin về các phân tử hoạt tính sinh học. Kết hợp với RDKit, một công cụ mạnh mẽ để tính toán các thuộc tính phân tử, bạn có thể xây dựng các mô hình dự đoán và khám phá các mối quan hệ quan trọng trong quá trình phát triển thuốc.

**Mô hình phân tích đề xuất:**

* **Phân tích mô tả (Descriptive Analysis):** Thống kê mô tả các thuộc tính hóa lý của các hợp chất (ví dụ: trọng lượng phân tử, LogP, số lượng vòng, v.v.) và hoạt tính sinh học của chúng (ví dụ: IC50, Ki). Phân tích này giúp bạn hiểu rõ hơn về dữ liệu và xác định các xu hướng tiềm năng.
* **Phân tích tương quan (Correlation Analysis):** Tìm kiếm mối tương quan giữa các thuộc tính hóa lý và hoạt tính sinh học. Ví dụ, bạn có thể tìm hiểu xem trọng lượng phân tử có tương quan với IC50 hay không.
* **Mô hình hóa QSAR/QSPR (QSAR/QSPR Modeling):** Xây dựng các mô hình định lượng mối quan hệ cấu trúc-hoạt tính (QSAR) hoặc cấu trúc-tính chất (QSPR). Các mô hình này có thể dự đoán hoạt tính sinh học hoặc tính chất của các hợp chất mới dựa trên cấu trúc của chúng.
* **Phân cụm (Clustering):** Phân loại các hợp chất thành các nhóm dựa trên sự tương đồng về cấu trúc hoặc hoạt tính. Điều này có thể giúp bạn xác định các họ hợp chất tiềm năng hoặc các khung cấu trúc quan trọng.

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

**2.1. Chuẩn bị dữ liệu (Data Preparation)**

* **Tiếng Việt:** Sử dụng pgAdmin để chạy các truy vấn SQL trên cơ sở dữ liệu ChEMBL 35. Lưu kết quả truy vấn vào các file CSV trong thư mục ../data/.
* **English:** Use pgAdmin to run SQL queries on the ChEMBL 35 database. Save the query results to CSV files in the ../data/ directory.

**2.2. Phân tích dữ liệu với Python (Data Analysis with Python)**

* **Tiếng Việt:** Sử dụng Jupyter Notebook để viết code Python. Sử dụng thư viện RDKit để tính toán các thuộc tính phân tử. Sử dụng các thư viện như pandas, numpy, scikit-learn để phân tích dữ liệu và xây dựng mô hình.
* **English:** Use Jupyter Notebook to write Python code. Use the RDKit library to calculate molecular properties. Use libraries like pandas, numpy, and scikit-learn to analyze data and build models.

**3. Code SQL và Python (SQL and Python Code)**

**3.1. Code SQL (SQL Code)**

-- Lấy 100 dòng dữ liệu từ bảng activities và molecules  
-- Get 100 rows of data from the activities and molecules tables  
SELECT act.activity\_id,  
 mol.molecule\_chembl\_id,  
 act.standard\_type,  
 act.standard\_relation,  
 act.standard\_value,  
 act.standard\_units,  
 mol.pref\_name  
FROM activities act  
JOIN molecule\_dictionary mol ON act.molregno = mol.molregno  
WHERE act.standard\_type = 'IC50' -- Lọc theo loại hoạt tính IC50  
 AND act.standard\_relation = '=' -- Lọc theo quan hệ '='  
 AND act.standard\_value IS NOT NULL -- Loại bỏ giá trị NULL  
 AND act.standard\_value > 0 -- Loại bỏ giá trị âm hoặc bằng 0  
 AND act.standard\_units = 'nM' -- Lọc theo đơn vị nM  
LIMIT 100;

**3.2. Code Python (Python Code)**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import numpy as np  
import os  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
  
# Đường dẫn cơ sở của dự án  
# Base path of the project  
base\_path = ".."  
  
# Đường dẫn đến file CSV chứa dữ liệu  
# Path to the CSV file containing the data  
data\_path = os.path.join(base\_path, "data", "chembl\_ic50\_data.csv")  
  
# Đọc dữ liệu từ file CSV  
# Read data from CSV file  
df = pd.read\_csv(data\_path)  
  
# Hiển thị thông tin cơ bản về dữ liệu  
# Display basic information about the data  
print(df.head())  
  
# Hàm tính toán các thuộc tính phân tử sử dụng RDKit  
# Function to calculate molecular properties using RDKit  
def calculate\_properties(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 mw = Descriptors.MolWt(mol)  
 logp = Descriptors.MolLogP(mol)  
 num\_hba = Descriptors.NumHAcceptors(mol)  
 num\_hbd = Descriptors.NumHDonors(mol)  
 return mw, logp, num\_hba, num\_hbd  
 else:  
 return None, None, None, None  
  
# Áp dụng hàm tính toán thuộc tính cho cột 'canonical\_smiles'  
# Apply the property calculation function to the 'canonical\_smiles' column  
df[['mw', 'logp', 'num\_hba', 'num\_hbd']] = df['canonical\_smiles'].apply(lambda x: pd.Series(calculate\_properties(x)))  
  
# Loại bỏ các dòng có giá trị NaN  
# Remove rows with NaN values  
df = df.dropna()  
  
# Chuyển đổi cột 'standard\_value' thành kiểu số  
# Convert the 'standard\_value' column to numeric type  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'])  
  
# Chuẩn bị dữ liệu cho mô hình hóa  
# Prepare data for modeling  
X = df[['mw', 'logp', 'num\_hba', 'num\_hbd']]  
y = df['standard\_value']  
  
# Chia dữ liệu thành tập huấn luyện và tập kiểm tra  
# 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)  
  
# Xây dựng mô hình hồi quy tuyến tính  
# Build a linear regression model  
model = LinearRegression()  
  
# Huấn luyện mô hình trên tập huấn luyện  
# Train the model on the training set  
model.fit(X\_train, y\_train)  
  
# Dự đoán trên tập kiểm tra  
# Predict on the test set  
y\_pred = model.predict(X\_test)  
  
# Đánh giá mô hình bằng MSE  
# Evaluate the model using MSE  
mse = mean\_squared\_error(y\_test, y\_pred)  
print(f"Mean Squared Error: {mse}")

**4. Ví dụ code SQL và Python mẫu (Sample SQL and Python Code Examples)**

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

**Ví dụ 1: Lọc các hợp chất có trọng lượng phân tử trong khoảng nhất định (Filtering compounds by molecular weight range)**

* **SQL:**

SELECT mol.molecule\_chembl\_id,  
 mol.pref\_name,  
 cp.mw\_freebase  
FROM molecule\_dictionary mol  
JOIN compound\_properties cp ON mol.molregno = cp.molregno  
WHERE cp.mw\_freebase BETWEEN 200 AND 500  
LIMIT 100;

* **Python:**

import pandas as pd  
  
# Giả sử dữ liệu đã được đọc vào DataFrame 'df'  
# Assuming data has been read into DataFrame 'df'  
  
# Lọc các hợp chất có trọng lượng phân tử trong khoảng 200-500  
# Filter compounds with molecular weight between 200-500  
df\_filtered = df[(df['mw'] >= 200) & (df['mw'] <= 500)]  
  
print(df\_filtered.head())

**Ví dụ 2: Tính số lượng hợp chất cho mỗi loại hoạt tính (Counting compounds for each activity type)**

* **SQL:**

SELECT standard\_type,  
 COUNT(\*) AS num\_compounds  
FROM activities  
GROUP BY standard\_type  
ORDER BY num\_compounds DESC  
LIMIT 10;

* **Python:**

# Tính số lượng hợp chất cho mỗi loại hoạt tính  
# Count the number of compounds for each activity type  
activity\_counts = df['standard\_type'].value\_counts()  
  
print(activity\_counts.head(10))

**Ví dụ 3: Tìm các hợp chất có hoạt tính cao (Finding highly active compounds)**

* **SQL:**

SELECT act.activity\_id,  
 mol.molecule\_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\_relation = '='  
 AND act.standard\_value < 100 -- IC50 < 100 nM  
LIMIT 100;

* **Python:**

# Lọc các hợp chất có IC50 < 100 nM  
# Filter compounds with IC50 < 100 nM  
df\_active = df[(df['standard\_type'] == 'IC50') & (df['standard\_relation'] == '=') & (df['standard\_value'] < 100)]  
  
print(df\_active.head())

**Ví dụ 4: Tính toán các thuộc tính phân tử sử dụng RDKit (Calculating molecular properties using RDKit)**

* **SQL:** (Không áp dụng, vì SQL không thể tính toán các thuộc tính phân tử phức tạp như RDKit)
* **Python:**

from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def calculate\_logp(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return Descriptors.MolLogP(mol)  
 else:  
 return None  
  
# Áp dụng hàm tính toán LogP cho cột 'canonical\_smiles'  
# Apply the LogP calculation function to the 'canonical\_smiles' column  
df['logp'] = df['canonical\_smiles'].apply(calculate\_logp)

**Ví dụ 5: Xây dựng mô hình hồi quy tuyến tính đơn giản (Building a simple linear regression model)**

* **SQL:** (Không áp dụng, vì SQL không phù hợp cho việc xây dựng mô hình máy học)
* **Python:**

from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import mean\_squared\_error  
  
# Chuẩn bị dữ liệu  
# Prepare data  
X = df[['mw']] # Sử dụng trọng lượng phân tử làm biến độc lập  
y = df['standard\_value'] # Sử dụng IC50 làm biến phụ thuộc  
  
# Chia dữ liệu thành tập huấn luyện và tập kiểm tra  
# 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)  
  
# Xây dựng mô hình  
# Build the model  
model = LinearRegression()  
  
# Huấn luyện mô hình  
# Train the model  
model.fit(X\_train, y\_train)  
  
# Dự đoán  
# Predict  
y\_pred = model.predict(X\_test)  
  
# Đánh giá  
# Evaluate  
mse = mean\_squared\_error(y\_test, y\_pred)  
print(f"Mean Squared Error: {mse}")

**5. Lưu ý về lỗi (Error Notes)**

**5.1. Lỗi liên quan đến kiểu dữ liệu (Data Type Error):**

* **Lỗi:** ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard\_value ~ '^[0-9\.]+$'
* **Giải thích:** Lỗi này xảy ra do bạn đang cố gắng sử dụng toán tử ~ (regular expression matching) trên một cột kiểu số (numeric). Toán tử này chỉ hoạt động trên kiểu dữ liệu chuỗi (text).
* **Khắc phục:** Thay vì sử dụng ~, hãy sử dụng các hàm hoặc toán tử phù hợp để kiểm tra xem giá trị có phải là số hay không. Trong trường hợp này, bạn có thể bỏ qua điều kiện này vì bạn đã lọc act.standard\_value IS NOT NULL và act.standard\_value > 0.
* **English:** This error occurs because you are trying to use the ~ operator (regular expression matching) on a numeric column. This operator only works on string data types. To fix this, use appropriate functions or operators to check if the value is a number. In this case, you can skip this condition because you already filtered act.standard\_value IS NOT NULL and act.standard\_value > 0.

**5.2. Lỗi liên quan đến phiên bản scikit-learn (scikit-learn Version Error):**

* **Lỗi:** Phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean\_squared\_error.
* **Giải thích:** Tham số squared=False 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.
* **Khắc phục:** Có hai cách:
  1. **Nâng cấp scikit-learn:** Cập nhật phiên bản scikit-learn của bạn lên phiên bản mới nhất bằng lệnh pip install --upgrade scikit-learn.
  2. **Tính RMSE thủ công:** Nếu bạn không thể nâng cấp scikit-learn, bạn có thể tính RMSE bằng cách lấy căn bậc hai của MSE: rmse = np.sqrt(mse).
* **English:** The squared=False parameter was added to the mean\_squared\_error function in newer scikit-learn versions to return Root Mean Squared Error (RMSE) instead of MSE. To fix this, either upgrade your scikit-learn version using pip install --upgrade scikit-learn, or calculate RMSE manually by taking the square root of MSE: rmse = np.sqrt(mse).

Tôi hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn tiến xa hơn trong dự án của mình. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi!