# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Kết nối ChEMBL với DrugBank để tái định hướng thuốc

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 nghiên cứu và phát triển dược học, tôi sẽ cung cấp cho bạn hướng dẫn chi tiết, mã nguồn SQL và Python mẫu, cùng với những lưu ý quan trọng để đảm bảo dự án của bạn diễn ra suôn sẻ.

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

Dựa trên thông tin bạn cung cấp, chúng ta sẽ xây dựng một quy trình phân tích dữ liệu tuân theo các bước sau:

1. **Data Extraction (Trích xuất dữ liệu):** Sử dụng SQL để truy vấn cơ sở dữ liệu ChEMBL 35 trên PostgreSQL và lưu kết quả vào các file CSV.
2. **Data Preprocessing (Tiền xử lý dữ liệu):** Sử dụng Python (với thư viện RDKit và Pandas) để làm sạch, chuyển đổi và chuẩn hóa dữ liệu. Các bước có thể bao gồm:
   * Xử lý các giá trị missing.
   * Chuyển đổi các giá trị hoạt tính (IC50, Ki, EC50) sang pChEMBL.
   * Tính toán các đặc tính lý hóa (physicochemical properties) và fingerprint phân tử (molecular fingerprints) bằng RDKit.
3. **Exploratory Data Analysis (EDA) (Phân tích khám phá dữ liệu):** Sử dụng Python để khám phá dữ liệu, tìm kiếm các xu hướng và mối quan hệ giữa các biến. Các kỹ thuật có thể bao gồm:
   * Vẽ biểu đồ phân phối của các biến.
   * Tính toán các thống kê mô tả.
   * Phân tích tương quan.
4. **Model Building (Xây dựng mô hình):** Sử dụng các thuật toán học máy (ví dụ: Random Forest, Support Vector Machines) để xây dựng các mô hình dự đoán hoạt tính của các hợp chất.
5. **Model Evaluation (Đánh giá mô hình):** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp (ví dụ: RMSE, R-squared, AUC).

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

**2.1. Kết nối cơ sở dữ liệu (Connecting to the Database)**

**Tiếng Việt:**

Để kết nối đến cơ sở dữ liệu ChEMBL 35 của bạn, bạn cần sử dụng thư viện psycopg2 trong Python. Đảm bảo rằng bạn đã cài đặt thư viện này bằng lệnh pip install psycopg2.

**English:**

To connect to your ChEMBL 35 database, you need to use the psycopg2 library in Python. Make sure you have installed this library using the command pip install psycopg2.

import psycopg2  
  
# Database credentials  
db\_params = {  
 'host': '192.168.206.136',  
 'user': 'rd',  
 'password': 'rd',  
 'database': 'chembl\_35'  
}  
  
def connect\_to\_db(params):  
 """Connects to the PostgreSQL database."""  
 try:  
 conn = psycopg2.connect(\*\*params)  
 return conn  
 except psycopg2.Error as e:  
 print(f"Error connecting to the database: {e}")  
 return None  
  
# Establish connection  
connection = connect\_to\_db(db\_params)  
  
if connection:  
 print("Successfully connected to the database!")  
 # Perform database operations here  
 connection.close() # remember to close connection after done  
else:  
 print("Failed to connect to the database.")

**2.2. Trích xuất dữ liệu với SQL (Data Extraction with SQL)**

**Tiếng Việt:**

Sử dụng câu lệnh SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35. Lưu ý, để tránh lỗi “operator does not exist: numeric ~ unknown”, hãy sử dụng hàm CAST để chuyển đổi kiểu dữ liệu của cột standard\_value sang kiểu TEXT trước khi so sánh với regular expression.

**English:**

Use SQL queries to extract data from the ChEMBL 35 database. Note, to avoid the error “operator does not exist: numeric ~ unknown”, use the CAST function to convert the data type of the standard\_value column to TEXT before comparing with the regular expression.

-- Lấy 100 dòng dữ liệu  
-- Get 100 rows of data  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
WHERE act.standard\_type = 'IC50'  
 AND act.standard\_relation = '='  
 AND act.standard\_units = 'nM'  
 AND CAST(act.standard\_value AS TEXT) ~ '^[0-9\.]+$' -- fix error here  
LIMIT 100;

**2.3. Tiền xử lý dữ liệu với Python (Data Preprocessing with Python)**

**Tiếng Việt:**

Sử dụng thư viện Pandas để đọc dữ liệu từ file CSV, RDKit để tính toán fingerprint phân tử và pChEMBL, và Scikit-learn cho các bước tiền xử lý khác.

**English:**

Use the Pandas library to read data from CSV files, RDKit to calculate molecular fingerprints and pChEMBL, and Scikit-learn for other preprocessing steps.

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import AllChem  
import numpy as np  
  
def calculate\_pchembl(standard\_value):  
 """Calculates pChEMBL value."""  
 return -np.log10(standard\_value \* 1e-9)  
  
def calculate\_fingerprint(smiles):  
 """Calculates Morgan fingerprint using RDKit."""  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)  
 else:  
 return None  
  
# Example usage  
csv\_file\_path = '../data/your\_file.csv' # Replace with your actual file path  
df = pd.read\_csv(csv\_file\_path)  
  
# Calculate pChEMBL (assuming IC50 in nM)  
df['pchembl\_value'] = df['standard\_value'].apply(calculate\_pchembl)  
  
# Calculate molecular fingerprints  
df['mol'] = df['canonical\_smiles'].apply(Chem.MolFromSmiles)  
df = df[df['mol'].notna()]  
df['fingerprint'] = df['canonical\_smiles'].apply(calculate\_fingerprint)  
df = df[df['fingerprint'].notna()]  
  
print(df.head())

**2.4 Xử lý lỗi phiên bản Scikit-learn (Handling Scikit-learn Version Error)**

**Tiếng Việt:**

Nếu bạn gặp lỗi liên quan đến tham số squared=False trong hàm mean\_squared\_error, hãy kiểm tra phiên bản Scikit-learn của bạn. Nếu phiên bản cũ, bạn có thể nâng cấp lên phiên bản mới hơn hoặc bỏ tham số squared=False (khi đó, kết quả sẽ là MSE thay vì RMSE).

**English:**

If you encounter an error related to the squared=False parameter in the mean\_squared\_error function, check your Scikit-learn version. If the version is old, you can upgrade to a newer version or remove the squared=False parameter (in which case the result will be MSE instead of RMSE).

from sklearn.metrics import mean\_squared\_error  
  
# Example  
y\_true = [3, -0.5, 2, 7]  
y\_predicted = [2.5, 0.0, 2, 8]  
  
# Try to calculate RMSE; if it fails, calculate MSE instead  
try:  
 rmse = mean\_squared\_error(y\_true, y\_predicted, squared=False)  
 print(f"RMSE: {rmse}")  
except TypeError:  
 mse = mean\_squared\_error(y\_true, y\_predicted)  
 print(f"MSE: {mse}. Consider upgrading scikit-learn for RMSE support.")

**3. Ví dụ mã SQL và Python mẫu (Example SQL and Python Code Samples)**

**3.1. SQL Examples**

-- 1. Lấy danh sách các hợp chất có hoạt tính IC50 dưới 100 nM  
-- 1. Get a list of compounds with IC50 activity below 100 nM  
SELECT md.chembl\_id, cs.canonical\_smiles  
FROM molecule\_dictionary md  
JOIN compound\_structures cs ON md.molregno = cs.molregno  
JOIN activities act ON md.molregno = act.molregno  
WHERE act.standard\_type = 'IC50' AND act.standard\_value < 100 AND act.standard\_units = 'nM';  
  
-- 2. Lấy số lượng các hợp chất cho mỗi giá trị standard\_type  
-- 2. Get the number of compounds for each standard\_type  
SELECT act.standard\_type, COUNT(DISTINCT md.chembl\_id) AS num\_compounds  
FROM activities act  
JOIN molecule\_dictionary md ON act.molregno = md.molregno  
GROUP BY act.standard\_type;  
  
-- 3. Lấy các hợp chất có khối lượng phân tử nằm trong khoảng 400-500 Da  
-- 3. Get compounds with molecular weight between 400-500 Da  
SELECT md.chembl\_id, cs.canonical\_smiles  
FROM molecule\_dictionary md  
JOIN compound\_structures cs ON md.molregno = cs.molregno  
WHERE md.mw\_freebase BETWEEN 400 AND 500;  
  
-- 4. Lấy danh sách các mục tiêu (targets) liên quan đến một hợp chất cụ thể  
-- 4. Get a list of targets related to a specific compound  
SELECT td.chembl\_id, td.pref\_name  
FROM target\_dictionary td  
JOIN assays a ON td.tid = a.tid  
JOIN activities act ON a.assay\_id = act.assay\_id  
JOIN molecule\_dictionary md ON act.molregno = md.molregno  
WHERE md.chembl\_id = 'CHEMBL188'; -- Thay thế bằng Chembl ID bạn muốn  
  
-- 5. Lấy danh sách các protein có hoạt tính Ki  
-- 5. Get the list of proteins with Ki activities  
SELECT td.chembl\_id, td.pref\_name  
FROM target\_dictionary td  
JOIN assays a ON td.tid = a.tid  
JOIN activities act ON a.assay\_id = act.assay\_id  
WHERE act.standard\_type = 'Ki';

**3.2. Python Examples**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
# 1. Tính toán LogP cho một danh sách các SMILES  
# 1. Calculate LogP for a list of SMILES  
smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
logp\_values = []  
  
for smiles in smiles\_list:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 logp = Descriptors.MolLogP(mol)  
 logp\_values.append(logp)  
 else:  
 logp\_values.append(None)  
  
print(f"LogP values: {logp\_values}")  
  
# 2. Đọc dữ liệu từ CSV và tính toán số vòng thơm  
# 2. Read data from CSV and calculate the number of aromatic rings  
csv\_file = '../data/your\_file.csv'  
df = pd.read\_csv(csv\_file)  
  
def calculate\_aromatic\_rings(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Chem.Lipinski.NumAromaticRings(mol)  
 else:  
 return None  
  
df['aromatic\_rings'] = df['canonical\_smiles'].apply(calculate\_aromatic\_rings)  
print(df[['canonical\_smiles', 'aromatic\_rings']].head())  
  
# 3. Lọc các hợp chất tuân theo quy tắc 5 của Lipinski  
# 3. Filter compounds that follow Lipinski's rule of 5  
def lipinski\_rule\_of\_five(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 mw = Descriptors.MolWt(mol)  
 logp = Descriptors.MolLogP(mol)  
 hbd = Chem.Lipinski.NumHDonors(mol)  
 hba = Chem.Lipinski.NumHAcceptors(mol)  
 return mw <= 500 and logp <= 5 and hbd <= 5 and hba <= 10  
 else:  
 return False  
  
df['lipinski\_compliant'] = df['canonical\_smiles'].apply(lipinski\_rule\_of\_five)  
lipinski\_compliant\_df = df[df['lipinski\_compliant']]  
print(f"Number of Lipinski-compliant compounds: {len(lipinski\_compliant\_df)}")  
  
# 4. Lưu các fingerprints vào một file  
# 4. Save the fingerprints to a file  
from rdkit.Chem import MACCSkeys  
import pickle  
  
def calculate\_maccs\_keys(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return MACCSkeys.GenMACCSKeys(mol)  
 else:  
 return None  
  
df['maccs\_keys'] = df['canonical\_smiles'].apply(calculate\_maccs\_keys)  
maccs\_keys\_list = df['maccs\_keys'].tolist()  
  
with open('../data/maccs\_keys.pkl', 'wb') as f:  
 pickle.dump(maccs\_keys\_list, f)  
  
# 5. Tìm các cấu trúc con (substructure search)  
# 5. Find substructures  
from rdkit.Chem import Draw  
  
substructure\_smiles = 'c1ccccc1' # Benzen  
substructure = Chem.MolFromSmiles(substructure\_smiles)  
  
def has\_substructure(smiles, substructure):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return mol.HasSubstructMatch(substructure)  
 else:  
 return False  
  
df['has\_benzene'] = df['canonical\_smiles'].apply(lambda x: has\_substructure(x, substructure))  
benzene\_containing\_df = df[df['has\_benzene']]  
  
print(f"Number of compounds containing benzene: {len(benzene\_containing\_df)}")

**Lưu ý quan trọng (Important Notes):**

* **Đường dẫn file (File Paths):** Đảm bảo rằng các đường dẫn đến file CSV là chính xác. Sử dụng os.path.join(base\_path, ...) để tạo đường dẫn một cách linh hoạt.
* **Phiên bản thư viện (Library Versions):** Kiểm tra và cập nhật các thư viện (RDKit, Pandas, Scikit-learn) để đảm bảo tính tương thích và tận dụng các tính năng mới nhất.
* **Xử lý lỗi (Error Handling):** Thêm các khối try...except để xử lý các lỗi có thể xảy ra trong quá trình xử lý dữ liệu.
* **Bộ nhớ (Memory):** Với bộ dữ liệu lớn, hãy cân nhắc sử dụng các kỹ thuật như chunking (đọc dữ liệu theo từng phần) để giảm thiểu việc sử dụng bộ nhớ.

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