# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Sàng lọc chất điều biến epigenetic qua ChEMBL

Tuyệt vời! Với vai trò là một chuyên gia nghiên cứu và phát triển dược học, tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Dưới đây là phân tích, hướng dẫn và các ví dụ code theo yêu cầu của bạn, tập trung vào mã chủ đề Topic\_CheMBL\_35\_99.

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

Mục tiêu của chúng ta là khai thác dữ liệu ChEMBL 35 để hỗ trợ quá trình nghiên cứu và phát triển thuốc. Dựa trên mã chủ đề Topic\_CheMBL\_35\_99, chúng ta có thể tập trung vào các khía cạnh sau:

* **Phân tích Hoạt tính Sinh học (Bioactivity Analysis):**
  + Xác định các hợp chất có hoạt tính cao đối với một mục tiêu cụ thể (target).
  + Phân tích mối quan hệ cấu trúc-hoạt tính (SAR) để hiểu cách các thay đổi nhỏ trong cấu trúc phân tử ảnh hưởng đến hoạt tính.
  + Xây dựng mô hình dự đoán hoạt tính dựa trên cấu trúc (QSAR/QSPR).
* **Phân tích Tính chất Vật lý Hóa học (Physicochemical Property Analysis):**
  + Tính toán và phân tích các tính chất như LogP, độ tan, khối lượng phân tử, v.v.
  + Đánh giá khả năng hấp thụ, phân bố, chuyển hóa, thải trừ (ADMET) của các hợp chất.
* **Phân tích Cấu trúc Phân tử (Molecular Structure Analysis):**
  + Sử dụng RDKit để tạo fingerprints, descriptors và các biểu diễn cấu trúc khác.
  + Phân tích sự đa dạng cấu trúc của các hợp chất trong cơ sở dữ liệu.
  + Tìm kiếm các motif cấu trúc (scaffolds) phổ biến.

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

**English:**

This project aims to leverage ChEMBL 35 data for drug discovery and development. We’ll use PostgreSQL to query the database, RDKit for molecular manipulation, and Jupyter Notebooks for analysis and visualization. We’ll focus on analyzing bioactivity, physicochemical properties, and molecular structures of compounds in the database. The errors you encountered will be addressed in the code examples.

**Tiếng Việt:**

Dự án này nhằm mục đích tận dụng dữ liệu ChEMBL 35 cho quá trình khám phá và phát triển thuốc. Chúng ta sẽ sử dụng PostgreSQL để truy vấn cơ sở dữ liệu, RDKit để thao tác phân tử và Jupyter Notebooks để phân tích và trực quan hóa. Chúng ta sẽ tập trung vào việc phân tích hoạt tính sinh học, tính chất vật lý hóa học và cấu trúc phân tử của các hợp chất trong cơ sở dữ liệu. Các lỗi bạn gặp phải sẽ được giải quyết trong các ví dụ code.

**3. Code SQL & Python (SQL & Python Code)**

**a. SQL (để trích xuất dữ liệu từ ChEMBL 35):**

-- English: Retrieve 100 compounds with IC50 values for a specific target (e.g., CHEMBL205)  
-- Tiếng Việt: Lấy 100 hợp chất có giá trị IC50 cho một mục tiêu cụ thể (ví dụ: CHEMBL205)  
SELECT  
 cmp.chembl\_id,  
 act.standard\_value,  
 act.standard\_units,  
 md.molfile  
FROM  
 compound\_structures cmp  
JOIN  
 activities act ON cmp.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.chembl\_id = 'CHEMBL205' -- Replace with your target of interest  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9\.]+$' -- Corrected to handle numeric/text comparison  
LIMIT 100;

**Giải thích lỗi và sửa:**

* **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 so sánh một cột kiểu số (numeric) với một chuỗi (unknown).
* **Sửa:** Ép kiểu cột standard\_value về kiểu text bằng act.standard\_value::text.

**b. Python (để xử lý và phân tích dữ liệu):**

import os  
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 numpy as np # Import numpy  
  
# Define base path  
base\_path = "." # Replace with your actual base path  
  
# Load data from CSV (assuming you saved the SQL output to a CSV file)  
data\_path = os.path.join(base\_path, "data", "chembl\_205\_ic50\_100.csv") # Adjust filename  
df = pd.read\_csv(data\_path)  
  
# Convert standard\_value to numeric, handling potential errors  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce')  
df = df.dropna(subset=['standard\_value']) # Remove rows where conversion failed  
  
# Function to calculate molecular descriptors using RDKit  
def calculate\_descriptors(molfile):  
 mol = Chem.MolFromMolBlock(molfile)  
 if mol is None:  
 return None  
 descriptors = {}  
 descriptors['MolWt'] = Descriptors.MolWt(mol)  
 descriptors['LogP'] = Descriptors.MolLogP(mol)  
 # Add more Descriptors as needed  
 return descriptors  
  
# Apply the descriptor calculation to each molecule  
df['descriptors'] = df['molfile'].apply(calculate\_descriptors)  
  
# Handle cases where descriptor calculation fails  
df = df.dropna(subset=['descriptors'])  
  
# Convert descriptors to DataFrame and merge  
descriptors\_df = pd.DataFrame(df['descriptors'].tolist())  
df = pd.concat([df, descriptors\_df], axis=1)  
  
# Prepare data for modeling  
df = df.dropna(subset=['MolWt', 'LogP', 'standard\_value']) # Drop rows with NaN  
X = df[['MolWt', 'LogP']] # Features  
y = df['standard\_value'] # Target variable (IC50)  
  
# 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}")  
  
# English: Example of how to use the model to predict IC50 for a new compound  
# Tiếng Việt: Ví dụ về cách sử dụng mô hình để dự đoán IC50 cho một hợp chất mới  
new\_compound = pd.DataFrame({'MolWt': [300], 'LogP': [2.5]}) # Example values  
predicted\_ic50 = model.predict(new\_compound)[0]  
print(f"Predicted IC50 for new compound: {predicted\_ic50}")

**Giải thích và Sửa lỗi:**

* **Lỗi:** ValueError: scikit-learn version too old (liên quan đến squared=False trong mean\_squared\_error).
* **Giải thích:** Các phiên bản scikit-learn cũ yêu cầu squared=True (mặc định) hoặc không có tham số squared.
* **Sửa:** Trong các phiên bản mới, squared=False trả về Root Mean Squared Error (RMSE). Nếu bạn có phiên bản cũ, hãy bỏ qua tham số này hoặc cập nhật scikit-learn. Đoạn code trên đã được sửa để tương thích với mọi phiên bản bằng cách bỏ tham số squared.

**4. Ví dụ Code SQL và Python mẫu (Example SQL and Python Code)**

Dưới đây là 5 ví dụ code SQL và Python khác nhau, minh họa các trường hợp sử dụng phổ biến:

**Ví dụ 1: Lọc hợp chất dựa trên khoảng giá trị LogP**

* **SQL:**

-- English: Retrieve compounds with LogP values between 2 and 4  
-- Tiếng Việt: Lấy các hợp chất có giá trị LogP nằm trong khoảng từ 2 đến 4  
SELECT  
 cmp.chembl\_id,  
 props.value AS logp  
FROM  
 compound\_structures cmp  
JOIN  
 compound\_properties props ON cmp.molregno = props.molregno  
WHERE  
 props.property\_type = 'ALOGP'  
 AND props.value BETWEEN 2 AND 4  
LIMIT 100;

* **Python:**

# English: Load the data from the SQL query result (assuming it's in a CSV) and filter by LogP  
# Tiếng Việt: Tải dữ liệu từ kết quả truy vấn SQL (giả sử nó ở định dạng CSV) và lọc theo LogP  
import pandas as pd  
import os  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "compounds\_logp.csv") # Adjust filename  
df = pd.read\_csv(data\_path)  
df = df[(df['logp'] >= 2) & (df['logp'] <= 4)]  
print(df.head())

**Ví dụ 2: Tìm kiếm các hợp chất tương tự về mặt cấu trúc (Similarity Search)**

* **SQL:** (Không thể thực hiện trực tiếp trong SQL, cần sử dụng RDKit trong Python)
* **Python:**

# English: Calculate Morgan Fingerprints and find compounds similar to a given compound  
# Tiếng Việt: Tính toán Morgan Fingerprints và tìm các hợp chất tương tự với một hợp chất cho trước  
from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.DataStructs import FingerprintSimilarity  
import pandas as pd  
import os  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "chembl\_compounds.csv") # Adjust filename  
df = pd.read\_csv(data\_path)  
  
# Assuming you have a column 'molfile' with the Molfile string  
def calculate\_morgan\_fingerprint(molfile):  
 mol = Chem.MolFromMolBlock(molfile)  
 if mol:  
 fp = AllChem.GetMorganFingerprint(mol, 2) # Radius 2  
 return fp  
 else:  
 return None  
  
df['fingerprint'] = df['molfile'].apply(calculate\_morgan\_fingerprint)  
df = df.dropna(subset=['fingerprint'])  
  
# Example: Choose a compound as a reference  
reference\_mol = Chem.MolFromMolBlock(df['molfile'].iloc[0]) # First compound as reference  
reference\_fp = AllChem.GetMorganFingerprint(reference\_mol, 2)  
  
# Calculate similarity to the reference compound  
def calculate\_similarity(fp):  
 return FingerprintSimilarity(reference\_fp, fp)  
  
df['similarity'] = df['fingerprint'].apply(calculate\_similarity)  
  
# Sort by similarity  
df\_sorted = df.sort\_values(by='similarity', ascending=False)  
print(df\_sorted.head())

**Ví dụ 3: Tính toán số lượng các vòng thơm (Number of Aromatic Rings)**

* **SQL:** (Không thể thực hiện trực tiếp trong SQL, cần sử dụng RDKit trong Python)
* **Python:**

# English: Calculate the number of aromatic rings in each compound  
# Tiếng Việt: Tính toán số lượng vòng thơm trong mỗi hợp chất  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import pandas as pd  
import os  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "chembl\_compounds.csv") # Adjust filename  
df = pd.read\_csv(data\_path)  
  
def calculate\_aromatic\_rings(molfile):  
 mol = Chem.MolFromMolBlock(molfile)  
 if mol:  
 return Descriptors.NumAromaticRings(mol)  
 else:  
 return None  
  
df['aromatic\_rings'] = df['molfile'].apply(calculate\_aromatic\_rings)  
print(df.head())

**Ví dụ 4: Phân tích phân bố hoạt tính (Activity Distribution Analysis)**

* **SQL:**

-- English: Get the distribution of IC50 values for a specific target  
-- Tiếng Việt: Lấy phân bố giá trị IC50 cho một mục tiêu cụ thể  
SELECT  
 act.standard\_value  
FROM  
 activities act  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.chembl\_id = 'CHEMBL205' -- Replace with your target of interest  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9\.]+$'  
LIMIT 100;

* **Python:**

# English: Plot the distribution of IC50 values  
# Tiếng Việt: Vẽ biểu đồ phân bố giá trị IC50  
import pandas as pd  
import matplotlib.pyplot as plt  
import os  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "ic50\_values.csv") # Adjust filename  
df = pd.read\_csv(data\_path)  
  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce')  
df = df.dropna(subset=['standard\_value'])  
  
plt.hist(df['standard\_value'], bins=50)  
plt.xlabel("IC50 (nM)")  
plt.ylabel("Frequency")  
plt.title("Distribution of IC50 Values")  
plt.show()

**Ví dụ 5: Tìm các scaffold phổ biến (Common Scaffold Search)**

* **SQL:** (Không thể thực hiện trực tiếp trong SQL, cần sử dụng RDKit trong Python)
* **Python:**

# English: Find the most common molecular scaffolds  
# Tiếng Việt: Tìm các scaffold phân tử phổ biến nhất  
from rdkit import Chem  
from rdkit.Chem.Scaffolds import MurckoScaffold  
from collections import Counter  
import pandas as pd  
import os  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "chembl\_compounds.csv") # Adjust filename  
df = pd.read\_csv(data\_path)  
  
def get\_murcko\_scaffold(molfile):  
 mol = Chem.MolFromMolBlock(molfile)  
 if mol:  
 scaffold = MurckoScaffold.GetScaffoldForMol(mol)  
 return Chem.MolToSmiles(scaffold)  
 else:  
 return None  
  
df['scaffold'] = df['molfile'].apply(get\_murcko\_scaffold)  
df = df.dropna(subset=['scaffold'])  
  
scaffold\_counts = Counter(df['scaffold'])  
most\_common\_scaffolds = scaffold\_counts.most\_common(10)  
  
print("Most Common Scaffolds:")  
print(most\_common\_scaffolds)

**Lưu ý quan trọng:**

* Thay đổi đường dẫn file (data\_path) và tên file cho phù hợp với cấu trúc thư mục và tên file thực tế của bạn.
* Điều chỉnh các thông số như tên mục tiêu (CHEMBL ID), loại hoạt tính (IC50), đơn vị (nM) cho phù hợp với mục tiêu nghiên cứu của bạn.
* Đây chỉ là những ví dụ cơ bản. Bạn có thể mở rộng chúng bằng cách thêm nhiều descriptors, sử dụng các thuật toán học máy phức tạp hơn, và thực hiện các phân tích chuyên sâu hơn.
* Luôn kiểm tra và làm sạch dữ liệu trước khi phân tích.

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