# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Vẽ phân bố IC50 trên các đích tác dụng

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 hiểu rõ tầm quan trọng của việc khai thác dữ liệu ChEMBL để tìm kiếm và phát triển các hợp chất tiềm năng. Dưới đây là phân tích chi tiết, hướng dẫn song ngữ và các ví dụ code mẫu để bạn tham khảo:

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

* **Mục tiêu (Objective):** Dự án này nhằm mục đích phân tích dữ liệu từ cơ sở dữ liệu ChEMBL 35, sử dụng RDKit để xử lý thông tin hóa học và trích xuất các đặc trưng (features) hữu ích. Từ đó, chúng ta có thể xây dựng các mô hình dự đoán hoạt tính sinh học, tìm kiếm các mối tương quan giữa cấu trúc và hoạt tính (SAR), hoặc xác định các “scaffold” tiềm năng cho việc phát triển thuốc.
* **Dữ liệu (Data):** Dữ liệu chính đến từ cơ sở dữ liệu ChEMBL 35, bao gồm thông tin về cấu trúc hóa học (SMILES), hoạt tính sinh học (IC50, Ki, EC50,…), và các thuộc tính khác của các hợp chất.
* **Công cụ (Tools):**
  + **PostgreSQL (psql):** Hệ quản trị cơ sở dữ liệu quan hệ để truy vấn và trích xuất dữ liệu từ ChEMBL 35.
  + **RDKit:** Thư viện Python mã nguồn mở để xử lý thông tin hóa học, tính toán các descriptor phân tử (ví dụ: trọng lượng phân tử, logP, số lượng vòng,…) và tạo fingerprint.
  + **Jupyter Notebook:** Môi trường lập trình tương tác để viết và chạy code Python, trực quan hóa dữ liệu và ghi lại quá trình phân tích.
  + **Scikit-learn:** Thư viện Python để xây dựng và đánh giá các mô hình học máy.
* **Quy trình (Workflow):**
  1. **Trích xuất dữ liệu (Data Extraction):** Sử dụng SQL để truy vấn và trích xuất dữ liệu cần thiết từ cơ sở dữ liệu ChEMBL 35 (ví dụ: các hợp chất có hoạt tính trên một target cụ thể). Lưu kết quả vào file CSV.
  2. **Tiền xử lý dữ liệu (Data Preprocessing):**
     + Đọc dữ liệu từ file CSV vào Pandas DataFrame.
     + Làm sạch dữ liệu (ví dụ: loại bỏ các giá trị NaN, chuẩn hóa các đơn vị đo).
     + Chuyển đổi SMILES thành các đối tượng RDKit Molecule.
  3. **Tính toán Descriptor (Descriptor Calculation):** Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc hóa học.
  4. **Xây dựng Mô hình (Model Building):** Sử dụng Scikit-learn để xây dựng các mô hình học máy (ví dụ: Random Forest, SVM, Linear Regression) để dự đoán hoạt tính sinh học.
  5. **Đánh giá Mô hình (Model Evaluation):** Đánh giá hiệu năng của mô hình bằng các metric phù hợp (ví dụ: R-squared, RMSE, AUC).
  6. **Phân tích Kết quả (Result Analysis):** Phân tích các kết quả dự đoán, xác định các hợp chất tiềm năng và các yếu tố cấu trúc quan trọng ảnh hưởng đến hoạt tính.

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

Dưới đây là hướng dẫn chi tiết các bước thực hiện, kèm theo giải thích song ngữ Anh-Việt:

* **Kết nối đến cơ sở dữ liệu ChEMBL (Connect to ChEMBL database):**
  + **Tiếng Anh:** Use the provided IP address, username, password, and database name to connect to the ChEMBL 35 database using pgAdmin.
  + **Tiếng Việt:** Sử dụng địa chỉ IP, tên người dùng, mật khẩu và tên cơ sở dữ liệu đã cung cấp để kết nối đến cơ sở dữ liệu ChEMBL 35 bằng pgAdmin.
* **Viết truy vấn SQL (Write SQL queries):**
  + **Tiếng Anh:** Write SQL queries to extract the necessary data from the ChEMBL database. For example, you can select compounds with specific activity values against a target of interest.
  + **Tiếng Việt:** Viết các truy vấn SQL để trích xuất dữ liệu cần thiết từ cơ sở dữ liệu ChEMBL. Ví dụ, bạn có thể chọn các hợp chất có giá trị hoạt tính cụ thể đối với một mục tiêu quan tâm.
* **Lưu dữ liệu vào file CSV (Save data to CSV files):**
  + **Tiếng Anh:** Execute the SQL queries in pgAdmin and save the results as CSV files in the ../data/ directory.
  + **Tiếng Việt:** Thực thi các truy vấn SQL trong pgAdmin và lưu kết quả dưới dạng file CSV trong thư mục ../data/.
* **Đọc dữ liệu trong Jupyter Notebook (Read data in Jupyter Notebook):**
  + **Tiếng Anh:** Use Pandas to read the CSV files into DataFrames in your Jupyter Notebook.
  + **Tiếng Việt:** Sử dụng Pandas để đọc các file CSV vào DataFrames trong Jupyter Notebook của bạn.
* **Xử lý dữ liệu bằng RDKit (Process data with RDKit):**
  + **Tiếng Anh:** Use RDKit to process the SMILES strings, calculate molecular descriptors, and generate fingerprints.
  + **Tiếng Việt:** Sử dụng RDKit để xử lý chuỗi SMILES, tính toán các descriptor phân tử và tạo fingerprint.
* **Xây dựng và đánh giá mô hình (Build and evaluate models):**
  + **Tiếng Anh:** Use Scikit-learn to build machine learning models and evaluate their performance.
  + **Tiếng Việt:** Sử dụng Scikit-learn để xây dựng các mô hình học máy và đánh giá hiệu suất của chúng.

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

* **SQL:**

-- Lấy 100 hợp chất có hoạt tính IC50 dưới 1000 nM trên target CHEMBL205 (Acetylcholinesterase)  
SELECT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 activities act  
JOIN  
 molecule\_dictionary md ON act.molregno = md.molregno  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
WHERE  
 act.standard\_type = 'IC50'  
 AND act.standard\_relation = '='  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value <= 1000  
 AND act.target\_chembl\_id = 'CHEMBL205'  
 AND cs.canonical\_smiles IS NOT NULL -- Loại bỏ SMILES rỗng  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Lọc các giá trị không hợp lệ  
LIMIT 100;

**Giải thích:**

* AND act.standard\_value ~ '^[0-9\.]+$' được thêm vào để lọc các giá trị standard\_value không phải là số. Toán tử ~ trong PostgreSQL thực hiện so khớp biểu thức chính quy (regular expression). Biểu thức '^[0-9\.]+$' đảm bảo rằng chuỗi chỉ chứa các chữ số (0-9) và dấu chấm (.), và phải có ít nhất một ký tự. Điều này giúp tránh lỗi khi chuyển đổi sang kiểu số.
* **Python:**

import pandas as pd  
import os  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error, r2\_score  
import numpy as np # Import numpy  
  
# Đường dẫn gốc của dự án  
base\_path = "." # Thay đổi nếu cần thiết  
  
# 1. Đọc dữ liệu từ file CSV  
data\_path = os.path.join(base\_path, "data", "chembl\_data.csv") # Thay "chembl\_data.csv" bằng tên file CSV của bạn  
df = pd.read\_csv(data\_path)  
  
# 2. Tiền xử lý dữ liệu  
df = df.dropna(subset=['canonical\_smiles', 'standard\_value']) # Loại bỏ các hàng có SMILES hoặc IC50 bị thiếu  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce') # Ép kiểu về số, lỗi sẽ thành NaN  
df = df.dropna(subset=['standard\_value']) # Loại bỏ các hàng có IC50 là NaN sau khi ép kiểu  
  
# Lọc các giá trị IC50 hợp lệ  
df = df[df['standard\_value'] > 0] # Loại bỏ IC50 <= 0  
  
# Giới hạn dữ liệu xuống 100 dòng  
df = df.head(100)  
  
# 3. Tính toán Descriptor  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 descriptors = {}  
 descriptors['MolLogP'] = Descriptors.MolLogP(mol)  
 descriptors['MolecularWeight'] = Descriptors.MolWt(mol)  
 descriptors['NumHAcceptors'] = Descriptors.NumHAcceptors(mol)  
 descriptors['NumHDonors'] = Descriptors.NumHDonors(mol)  
 return pd.Series(descriptors)  
  
df[['MolLogP', 'MolecularWeight', 'NumHAcceptors', 'NumHDonors']] = df['canonical\_smiles'].apply(calculate\_descriptors)  
df = df.dropna()  
  
# 4. Chuẩn bị dữ liệu cho mô hình  
X = df[['MolLogP', 'MolecularWeight', 'NumHAcceptors', 'NumHDonors']]  
y = np.log10(df['standard\_value']) # Chuyển đổi IC50 về thang log  
  
# 5. 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)  
  
# 6. Xây dựng và huấn luyện mô hình  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
# 7. Đánh giá mô hình  
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}")

**Giải thích:**

* **Import thư viện (Import Libraries):** Import các thư viện cần thiết như pandas, rdkit, sklearn.
* **Đọc dữ liệu (Read Data):** Đọc file CSV chứa dữ liệu ChEMBL vào DataFrame.
* **Tiền xử lý dữ liệu (Data Preprocessing):** Xử lý các giá trị bị thiếu, chuyển đổi SMILES thành đối tượng RDKit Molecule.
* **Tính toán Descriptor (Descriptor Calculation):** Tính toán các descriptor phân tử (ví dụ: MolLogP, MolecularWeight, NumHAcceptors, NumHDonors) sử dụng RDKit.
* **Xây dựng Mô hình (Model Building):** Xây dựng mô hình Random Forest để dự đoán hoạt tính sinh học (IC50).
* **Đánh giá Mô hình (Model Evaluation):** Đánh giá hiệu năng của mô hình bằng Mean Squared Error (MSE) và R-squared.

**Lưu ý về lỗi:**

* **ERROR: operator does not exist: numeric ~ unknown:** Lỗi này xảy ra khi bạn cố gắng sử dụng toán tử ~ (so khớp biểu thức chính quy) trên một cột kiểu số. Để khắc phục, bạn cần đảm bảo rằng cột act.standard\_value có kiểu dữ liệu text hoặc varchar. Nếu không, bạn có thể ép kiểu nó trước khi so sánh: sql AND CAST(act.standard\_value AS TEXT) ~ '^[0-9\.]+$'
* **squared=False trong mean\_squared\_error:** Nếu bạn đang sử dụng phiên bản Scikit-learn cũ, tham số squared=False có thể không được hỗ trợ. Trong trường hợp này, bạn có thể tính căn bậc hai của MSE để có được Root Mean Squared Error (RMSE): python mse = mean\_squared\_error(y\_test, y\_pred) rmse = mse\*\*0.5 print(f"Root Mean Squared Error: {rmse}")

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

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

**Ví dụ 1: Trích xuất dữ liệu và tính LogP**

* **SQL:**

-- Lấy CHEMBL\_ID, SMILES và LogP từ ChEMBL 35 cho 100 hợp chất  
SELECT md.chembl\_id, cs.canonical\_smiles  
FROM molecule\_dictionary md  
JOIN compound\_structures cs ON md.molregno = cs.molregno  
WHERE cs.canonical\_smiles IS NOT NULL  
LIMIT 100;

* **Python:**

import pandas as pd  
import os  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "chembl\_smiles.csv") # Thay "chembl\_smiles.csv" bằng tên file CSV của bạn  
df = pd.read\_csv(data\_path)  
  
def calculate\_logp(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 return Descriptors.MolLogP(mol)  
  
df['MolLogP'] = df['canonical\_smiles'].apply(calculate\_logp)  
print(df.head())

**Ví dụ 2: Lọc dữ liệu theo trọng lượng phân tử**

* **SQL:**

-- (Không thể lọc theo trọng lượng phân tử trực tiếp trong SQL, cần tính toán trước)  
-- Ví dụ này chỉ lấy SMILES và CHEMBL\_ID  
SELECT md.chembl\_id, cs.canonical\_smiles  
FROM molecule\_dictionary md  
JOIN compound\_structures cs ON md.molregno = cs.molregno  
WHERE cs.canonical\_smiles IS NOT NULL  
LIMIT 100;

* **Python:**

import pandas as pd  
import os  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "chembl\_smiles.csv") # Thay "chembl\_smiles.csv" bằng tên file CSV của bạn  
df = pd.read\_csv(data\_path)  
  
def calculate\_mw(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 return Descriptors.MolWt(mol)  
  
df['MolecularWeight'] = df['canonical\_smiles'].apply(calculate\_mw)  
df\_filtered = df[df['MolecularWeight'] <= 500] # Lọc các hợp chất có trọng lượng phân tử <= 500  
print(df\_filtered.head())

**Ví dụ 3: Tính fingerprint và xây dựng mô hình đơn giản**

* **SQL:**

-- (Không thể tính fingerprint trong SQL, cần sử dụng Python)  
-- Ví dụ này chỉ lấy SMILES và CHEMBL\_ID  
SELECT md.chembl\_id, cs.canonical\_smiles  
FROM molecule\_dictionary md  
JOIN compound\_structures cs ON md.molregno = cs.molregno  
WHERE cs.canonical\_smiles IS NOT NULL  
LIMIT 100;

* **Python:**

import pandas as pd  
import os  
from rdkit import Chem  
from rdkit.Chem import AllChem  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "chembl\_smiles.csv") # Thay "chembl\_smiles.csv" bằng tên file CSV của bạn  
df = pd.read\_csv(data\_path)  
  
def calculate\_fingerprint(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=1024)  
 return np.array(fp)  
  
df['Fingerprint'] = df['canonical\_smiles'].apply(calculate\_fingerprint)  
df = df.dropna(subset=['Fingerprint'])  
  
# Tạo dữ liệu giả định cho hoạt tính (ví dụ)  
df['Activity'] = np.random.randint(0, 2, df.shape[0]) # 0 hoặc 1  
  
X = np.vstack(df['Fingerprint'].values)  
y = df['Activity'].values  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = LogisticRegression()  
model.fit(X\_train, y\_train)  
  
y\_pred = model.predict(X\_test)  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"Accuracy: {accuracy}")

**Ví dụ 4: Tìm kiếm các motif phổ biến**

* **SQL:**

-- (Không thể tìm kiếm motif trực tiếp trong SQL, cần sử dụng Python)  
-- Ví dụ này chỉ lấy SMILES và CHEMBL\_ID  
SELECT md.chembl\_id, cs.canonical\_smiles  
FROM molecule\_dictionary md  
JOIN compound\_structures cs ON md.molregno = cs.molregno  
WHERE cs.canonical\_smiles IS NOT NULL  
LIMIT 100;

* **Python:**

import pandas as pd  
import os  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from rdkit.Chem import rdFMCS  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "chembl\_smiles.csv") # Thay "chembl\_smiles.csv" bằng tên file CSV của bạn  
df = pd.read\_csv(data\_path)  
  
mols = [Chem.MolFromSmiles(s) for s in df['canonical\_smiles'] if Chem.MolFromSmiles(s) is not None]  
  
# Tìm kiếm motif chung lớn nhất (MCS)  
mcs = rdFMCS.FindMCS(mols)  
mcs\_mol = Chem.MolFromSmarts(mcs.smartsString)  
  
print(mcs.smartsString) # In ra cấu trúc MCS

**Ví dụ 5: Phân tích SAR đơn giản (Structure-Activity Relationship)**

* **SQL:**

-- Lấy CHEMBL\_ID, SMILES và IC50 từ ChEMBL 35 cho 100 hợp chất có IC50  
SELECT md.chembl\_id, cs.canonical\_smiles, act.standard\_value  
FROM molecule\_dictionary md  
JOIN compound\_structures cs ON md.molregno = cs.molregno  
JOIN activities act ON md.molregno = act.molregno  
WHERE cs.canonical\_smiles IS NOT NULL  
AND act.standard\_type = 'IC50'  
AND act.standard\_units = 'nM'  
AND act.standard\_relation = '='  
LIMIT 100;

* **Python:**

import pandas as pd  
import os  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data", "chembl\_ic50.csv") # Thay "chembl\_ic50.csv" bằng tên file CSV của bạn  
df = pd.read\_csv(data\_path)  
  
def calculate\_logp(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 return Descriptors.MolLogP(mol)  
  
df['MolLogP'] = df['canonical\_smiles'].apply(calculate\_logp)  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Chuyển IC50 sang pIC50  
  
# In ra mối tương quan giữa LogP và pIC50  
print(df[['MolLogP', 'pIC50']].corr())

**Lưu ý quan trọng:**

* Hãy nhớ thay đổi đường dẫn file và tên file CSV cho phù hợp với dự án của bạn.
* Các ví dụ trên chỉ là điểm khởi đầu. Bạn có thể tùy chỉnh và mở rộng chúng để phù hợp với mục tiêu nghiên cứu cụ thể của bạn.
* Việc phân tích SAR đòi hỏi kiến thức sâu về hóa học và sinh học. Hãy tham khảo các tài liệu khoa học và tư vấn với các chuyên gia để có được kết quả chính xác và ý nghĩa.

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