# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Ánh xạ protein giữa người và chuột

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 song ngữ, code mẫu SQL và Python để giúp bạn bắt đầu, đồng thời giải quyết các vấn đề bạn đã đề cập.

**Chủ đề: Topic\_CheMBL\_35\_28**

**Tên chủ đề: Phân tích mối tương quan giữa các đặc tính lý hóa (Physicochemical Properties) và hoạt tính sinh học (Biological Activity) của các hợp chất trong ChEMBL 35.**

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

Mục tiêu của chủ đề này là tìm hiểu xem các đặc tính lý hóa của một phân tử (ví dụ: trọng lượng phân tử, logP, số lượng liên kết hydro, v.v.) ảnh hưởng như thế nào đến hoạt tính sinh học của nó (ví dụ: IC50, Ki). Mô hình phân tích sẽ bao gồm các bước sau:

* **Thu thập và tiền xử lý dữ liệu:** Lấy dữ liệu từ cơ sở dữ liệu ChEMBL 35, làm sạch và chuyển đổi dữ liệu về hoạt tính sinh học (ví dụ: chuyển đổi IC50 sang pIC50).
* **Tính toán các đặc tính lý hóa:** Sử dụng RDKit để tính toán các đặc tính lý hóa cho mỗi phân tử.
* **Phân tích tương quan:** Sử dụng các phương pháp thống kê (ví dụ: hồi quy tuyến tính, hồi quy đa biến) để đánh giá mối tương quan giữa các đặc tính lý hóa và hoạt tính sinh học.
* **Xây dựng mô hình dự đoán:** (Tùy chọn) Xây dựng mô hình máy học để dự đoán hoạt tính sinh học dựa trên các đặc tính lý hóa.
* **Đánh giá mô hình:** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp (ví dụ: R-squared, RMSE).

**In English:**

The goal of this topic is to understand how the physicochemical properties of a molecule (e.g., molecular weight, logP, number of hydrogen bonds, etc.) affect its biological activity (e.g., IC50, Ki). The analytical model will include the following steps:

* **Data Collection and Preprocessing:** Retrieve data from the ChEMBL 35 database, clean, and transform the biological activity data (e.g., convert IC50 to pIC50).
* **Calculation of Physicochemical Properties:** Use RDKit to calculate physicochemical properties for each molecule.
* **Correlation Analysis:** Use statistical methods (e.g., linear regression, multivariate regression) to assess the correlation between physicochemical properties and biological activity.
* **Building a Predictive Model:** (Optional) Build a machine learning model to predict biological activity based on physicochemical properties.
* **Model Evaluation:** Evaluate the performance of the model using appropriate metrics (e.g., R-squared, RMSE).

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

**Bước 1: Kết nối đến cơ sở dữ liệu ChEMBL 35 (Connect to ChEMBL 35 Database):**

Sử dụng thông tin bạn cung cấp để kết nối đến cơ sở dữ liệu PostgreSQL.

import psycopg2  
  
conn = psycopg2.connect(  
 host="192.168.206.136",  
 user="rd",  
 password="rd",  
 database="chembl\_35"  
)  
  
cursor = conn.cursor()

**In English:**

Use the information you provided to connect to the PostgreSQL database.

**Bước 2: Truy vấn dữ liệu từ ChEMBL 35 (Query Data from ChEMBL 35):**

Sử dụng SQL để truy vấn dữ liệu cần thiết. **Lưu ý lỗi liên quan đến toán tử ~**. Toán tử ~ trong PostgreSQL là toán tử so sánh pattern matching (tương tự regex). Để khắc phục lỗi “ERROR: operator does not exist: numeric ~ unknown”, bạn cần đảm bảo rằng cột act.standard\_value phải là kiểu dữ liệu text trước khi sử dụng toán tử ~.

-- Lấy 100 dòng dữ liệu  
SELECT  
 cmp.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 compound\_structures cmp  
JOIN  
 activities act ON cmp.molregno = act.molregno  
WHERE act.standard\_type = 'IC50'  
AND act.standard\_units = 'nM'  
AND act.standard\_value IS NOT NULL  
AND act.standard\_value::text ~ '^[0-9\.]+$' -- Ép kiểu về text trước khi so sánh  
LIMIT 100;

**In English:**

Use SQL to query the necessary data. **Note the error related to the ~ operator.** The ~ operator in PostgreSQL is a pattern matching operator (similar to regex). To fix the error “ERROR: operator does not exist: numeric ~ unknown”, you need to ensure that the act.standard\_value column is of text data type before using the ~ operator. The ::text cast ensures this.

**Bước 3: Lưu dữ liệu vào file CSV (Save Data to CSV File):**

Sử dụng psycopg2 để lấy dữ liệu và lưu vào file CSV.

import csv  
import os  
  
base\_path = "../data" # Đảm bảo đường dẫn này đúng với cấu trúc thư mục của bạn  
  
sql\_query = """  
SELECT  
 cmp.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 compound\_structures cmp  
JOIN  
 activities act ON cmp.molregno = act.molregno  
WHERE 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;  
"""  
  
cursor.execute(sql\_query)  
results = cursor.fetchall()  
  
csv\_file\_path = os.path.join(base\_path, "chembl\_data.csv")  
  
with open(csv\_file\_path, 'w', newline='') as csvfile:  
 csv\_writer = csv.writer(csvfile)  
 csv\_writer.writerow([i[0] for i in cursor.description]) # Ghi header  
 csv\_writer.writerows(results)  
  
print(f"Dữ liệu đã được lưu vào: {csv\_file\_path}")  
  
conn.close()

**In English:**

Use psycopg2 to retrieve the data and save it to a CSV file.

**Bước 4: Đọc dữ liệu từ file CSV và tính toán đặc tính lý hóa bằng RDKit (Read Data from CSV File and Calculate Physicochemical Properties using RDKit):**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import numpy as np  
  
# Đọc dữ liệu từ file CSV  
csv\_file\_path = os.path.join(base\_path, "chembl\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Hàm tính toán đặc tính lý hóa  
def calculate\_physicochemical\_properties(chembl\_id):  
 try:  
 # Lấy SMILES từ ChEMBL (cần một truy vấn SQL khác)  
 sql\_query = f"""  
 SELECT cs.canonical\_smiles  
 FROM compound\_structures cs  
 WHERE cs.chembl\_id = '{chembl\_id}'  
 """  
 cursor = conn.cursor()  
 cursor.execute(sql\_query)  
 smiles = cursor.fetchone()[0]  
  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
  
 # Tính toán các đặc tính lý hóa  
 mw = Descriptors.MolWt(mol)  
 logp = Chem.Crippen.MolLogP(mol)  
 hbd = Chem.rdMolDescriptors.CalcNumHBD(mol)  
 hba = Chem.rdMolDescriptors.CalcNumHBA(mol)  
 return mw, logp, hbd, hba  
 except Exception as e:  
 print(f"Lỗi khi tính toán đặc tính cho {chembl\_id}: {e}")  
 return None  
  
# Áp dụng hàm tính toán đặc tính lý hóa cho mỗi hợp chất  
df[['molecular\_weight', 'logp', 'hbd', 'hba']] = df['chembl\_id'].apply(lambda x: pd.Series(calculate\_physicochemical\_properties(x)))  
  
# Chuyển đổi standard\_value sang pIC50  
df = df.dropna(subset=['standard\_value'])  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'])  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Chuyển đổi nM sang M  
  
# In ra DataFrame  
print(df.head())

**In English:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import numpy as np  
  
# Read data from CSV file  
csv\_file\_path = os.path.join(base\_path, "chembl\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Function to calculate physicochemical properties  
def calculate\_physicochemical\_properties(chembl\_id):  
 try:  
 # Get SMILES from ChEMBL (requires another SQL query)  
 sql\_query = f"""  
 SELECT cs.canonical\_smiles  
 FROM compound\_structures cs  
 WHERE cs.chembl\_id = '{chembl\_id}'  
 """  
 cursor = conn.cursor()  
 cursor.execute(sql\_query)  
 smiles = cursor.fetchone()[0]  
  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
  
 # Calculate physicochemical properties  
 mw = Descriptors.MolWt(mol)  
 logp = Chem.Crippen.MolLogP(mol)  
 hbd = Chem.rdMolDescriptors.CalcNumHBD(mol)  
 hba = Chem.rdMolDescriptors.CalcNumHBA(mol)  
 return mw, logp, hbd, hba  
 except Exception as e:  
 print(f"Error calculating properties for {chembl\_id}: {e}")  
 return None  
  
# Apply the function to each compound  
df[['molecular\_weight', 'logp', 'hbd', 'hba']] = df['chembl\_id'].apply(lambda x: pd.Series(calculate\_physicochemical\_properties(x)))  
  
# Convert standard\_value to pIC50  
df = df.dropna(subset=['standard\_value'])  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'])  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Convert nM to M  
  
# Print the DataFrame  
print(df.head())

**Bước 5: Phân tích tương quan (Correlation Analysis):**

# Tính toán ma trận tương quan  
correlation\_matrix = df[['pIC50', 'molecular\_weight', 'logp', 'hbd', 'hba']].corr()  
print(correlation\_matrix)  
  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.heatmap(correlation\_matrix, annot=True)  
plt.show()

**In English:**

# Calculate the correlation matrix  
correlation\_matrix = df[['pIC50', 'molecular\_weight', 'logp', 'hbd', 'hba']].corr()  
print(correlation\_matrix)  
  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.heatmap(correlation\_matrix, annot=True)  
plt.show()

**Bước 6: Xây dựng mô hình hồi quy (Regression Model):**

from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Chuẩn bị dữ liệu  
X = df[['molecular\_weight', 'logp', 'hbd', 'hba']].fillna(0) # Điền giá trị NaN bằng 0  
y = df['pIC50'].fillna(0)  
  
# 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)  
  
# Xây dựng mô hình hồi quy tuyến tính  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Dự đoán trên tập kiểm tra  
y\_pred = model.predict(X\_test)  
  
# 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}")

**In English:**

from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Prepare the data  
X = df[['molecular\_weight', 'logp', 'hbd', 'hba']].fillna(0) # Fill NaN values with 0  
y = df['pIC50'].fillna(0)  
  
# Split the 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)  
  
# Build a linear regression model  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Predict 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}")

**Lưu ý về lỗi sklearn:** Nếu bạn gặp lỗi liên quan đến squared=False trong mean\_squared\_error, hãy đảm bảo rằng bạn đang sử dụng phiên bản scikit-learn mới nhất. Nếu không, bạn có thể tính căn bậc hai của MSE để có RMSE.

**3. Code SQL và Python mẫu (Sample SQL and Python Code):**

Dưới đây là 5 ví dụ về code SQL và Python mẫu:

**Ví dụ 1: Lấy số lượng hợp chất cho mỗi loại hoạt tính (SQL):**

SELECT standard\_type, COUNT(\*) FROM activities GROUP BY standard\_type;

**In English:**

SELECT standard\_type, COUNT(\*) FROM activities GROUP BY standard\_type;

**Ví dụ 1: Lấy SMILES canonical và molecular weight của các chất (SQL, Python):**

SELECT  
 cmp.chembl\_id,  
 cs.canonical\_smiles  
FROM  
 compound\_structures cs join compound\_properties cmp  
 ON cs.molregno = cmp.molregno  
LIMIT 10;

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def get\_molecular\_weight(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Descriptors.MolWt(mol)  
 return None  
  
sql\_query = """  
SELECT  
 cmp.chembl\_id,  
 cs.canonical\_smiles  
FROM  
 compound\_structures cs join compound\_properties cmp  
 ON cs.molregno = cmp.molregno  
LIMIT 10;  
"""  
  
cursor.execute(sql\_query)  
results = cursor.fetchall()  
  
df = pd.DataFrame(results, columns=['chembl\_id', 'smiles'])  
df['molecular\_weight'] = df['smiles'].apply(get\_molecular\_weight)  
  
print(df)

**Ví dụ 2: Tính toán LogP cho một danh sách các chất (Python):**

from rdkit import Chem  
from rdkit.Chem import Crippen  
  
smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
  
for smiles in smiles\_list:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 logp = Crippen.MolLogP(mol)  
 print(f"LogP for {smiles}: {logp}")  
 else:  
 print(f"Invalid SMILES: {smiles}")

**In English:**

from rdkit import Chem  
from rdkit.Chem import Crippen  
  
smiles\_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']  
  
for smiles in smiles\_list:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 logp = Crippen.MolLogP(mol)  
 print(f"LogP for {smiles}: {logp}")  
 else:  
 print(f"Invalid SMILES: {smiles}")

**Ví dụ 3: Lọc các chất có trọng lượng phân tử nằm trong một khoảng nhất định (SQL, Python):**

-- Cần tạo một view hoặc bảng tạm để lưu trữ trọng lượng phân tử  
-- (Ví dụ này chỉ minh họa logic, cần điều chỉnh cho phù hợp với cấu trúc DB)  
SELECT chembl\_id FROM compound\_properties WHERE molecular\_weight > 200 AND molecular\_weight < 500;

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def calculate\_mw(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Descriptors.MolWt(mol)  
 return None  
  
sql\_query = """  
SELECT  
 cmp.chembl\_id,  
 cs.canonical\_smiles  
FROM  
 compound\_structures cs join compound\_properties cmp  
 ON cs.molregno = cmp.molregno  
LIMIT 100;  
"""  
  
cursor.execute(sql\_query)  
results = cursor.fetchall()  
  
df = pd.DataFrame(results, columns=['chembl\_id', 'smiles'])  
df['molecular\_weight'] = df['smiles'].apply(calculate\_mw)  
  
filtered\_df = df[(df['molecular\_weight'] > 200) & (df['molecular\_weight'] < 500)]  
print(filtered\_df)

**Ví dụ 4: Tính toán số lượng liên kết hydro cho và nhận (Python):**

from rdkit import Chem  
from rdkit.Chem import rdMolDescriptors  
  
smiles = 'CC(=O)Oc1ccccc1C(=O)O'  
mol = Chem.MolFromSmiles(smiles)  
  
if mol:  
 hbd = rdMolDescriptors.CalcNumHBD(mol)  
 hba = rdMolDescriptors.CalcNumHBA(mol)  
 print(f"Hydrogen Bond Donors: {hbd}")  
 print(f"Hydrogen Bond Acceptors: {hba}")  
else:  
 print("Invalid SMILES")

**In English:**

from rdkit import Chem  
from rdkit.Chem import rdMolDescriptors  
  
smiles = 'CC(=O)Oc1ccccc1C(=O)O'  
mol = Chem.MolFromSmiles(smiles)  
  
if mol:  
 hbd = rdMolDescriptors.CalcNumHBD(mol)  
 hba = rdMolDescriptors.CalcNumHBA(mol)  
 print(f"Hydrogen Bond Donors: {hbd}")  
 print(f"Hydrogen Bond Acceptors: {hba}")  
else:  
 print("Invalid SMILES")

**Ví dụ 5: Phân tích tương quan giữa LogP và hoạt tính (SQL, Python):**

**(SQL - Ý tưởng, cần điều chỉnh theo schema):**

-- Ví dụ này cần kết hợp dữ liệu từ nhiều bảng và có thể yêu cầu tạo view  
-- Mục đích là minh họa ý tưởng, cần điều chỉnh cho phù hợp với database schema  
-- SELECT corr(logp, pIC50) FROM ... JOIN ... WHERE ...;

import pandas as pd  
import numpy as np  
from rdkit import Chem  
from rdkit.Chem import Crippen  
  
# Giả sử bạn đã có DataFrame 'df' chứa SMILES và pIC50  
# Ví dụ:  
data = {'smiles': ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO'],  
 'pIC50': [6.5, 7.0, 5.5]}  
df = pd.DataFrame(data)  
  
# Tính LogP  
df['logp'] = df['smiles'].apply(lambda x: Crippen.MolLogP(Chem.MolFromSmiles(x)) if Chem.MolFromSmiles(x) else None)  
  
# Loại bỏ các hàng có giá trị LogP là None  
df = df.dropna(subset=['logp'])  
  
# Tính toán tương quan  
correlation = df['logp'].corr(df['pIC50'])  
print(f"Correlation between LogP and pIC50: {correlation}")

**Lưu ý quan trọng:**

* Thay thế các đường dẫn file và thông tin kết nối cơ sở dữ liệu bằng thông tin chính xác của bạn.
* Các truy vấn SQL cần được điều chỉnh để phù hợp với cấu trúc cơ sở dữ liệu ChEMBL 35.
* Bạn có thể cần cài đặt các thư viện Python cần thiết (ví dụ: psycopg2, rdkit, pandas, scikit-learn) bằng pip.
* Hãy nhớ đóng kết nối cơ sở dữ liệu sau khi hoàn thành.
* Để sử dụng lại các đối tượng cursor , hãy tạo lại đối tượng mới sau khi close.
* Với lượng dữ liệu lớn, hãy xem xét sử dụng các kỹ thuật tối ưu hóa hiệu suất.

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