# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Trực quan hóa mạng lưới target

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 này. Với vai trò là chuyên gia R&D dược học, tôi hiểu rõ tầm quan trọng của việc khai thác dữ liệu hóa sinh và hóa học để đẩy nhanh quá trình phát triển thuốc. Dưới đây là phân tích chi tiết, hướng dẫn song ngữ và các ví dụ code bạn có thể tham khảo:

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

Mục tiêu của bạn là phân tích dữ liệu ChEMBL 35 bằng cách kết hợp SQL (PostgreSQL) để truy vấn và trích xuất dữ liệu, RDKit để xử lý thông tin hóa học (ví dụ: tính toán fingerprints, descriptors), và Python (Jupyter Notebook) để phân tích thống kê và xây dựng mô hình.

**Mô hình phân tích đề xuất:**

* **Data Extraction & Cleaning (Trích xuất và làm sạch dữ liệu):**
  + Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35. Tập trung vào các bảng activities, molecule\_dictionary, compound\_structures, và các bảng liên quan khác.
  + Lọc dữ liệu dựa trên các tiêu chí cụ thể (ví dụ: loại mục tiêu, loại hoạt động, giá trị IC50, v.v.).
  + Xử lý các giá trị bị thiếu, loại bỏ các bản ghi trùng lặp và chuẩn hóa dữ liệu.
* **Feature Engineering (Xây dựng đặc trưng):**
  + Sử dụng RDKit để tính toán các đặc trưng hóa học từ cấu trúc phân tử (SMILES). Các đặc trưng phổ biến bao gồm:
    - **Fingerprints:** ECFP, FCFP, MACCS keys.
    - **Descriptors:** Lipinski’s Rule of 5 descriptors, tính chất vật lý hóa học (ví dụ: LogP, MW, TPSA).
  + Kết hợp các đặc trưng hóa học với các thông tin khác từ cơ sở dữ liệu (ví dụ: loại mục tiêu, loại hoạt động).
* **Exploratory Data Analysis (EDA) (Phân tích khám phá dữ liệu):**
  + Sử dụng Python (matplotlib, seaborn) để trực quan hóa dữ liệu và khám phá các mối quan hệ giữa các biến.
  + Thực hiện phân tích thống kê mô tả (ví dụ: tính trung bình, độ lệch chuẩn, phân phối) cho từng đặc trưng.
  + Xác định các đặc trưng quan trọng có ảnh hưởng đến hoạt tính sinh học.
* **Model Building & Evaluation (Xây dựng và đánh giá mô hình):**
  + Sử dụng các thuật toán học máy (ví dụ: Random Forest, Support Vector Machines, Neural Networks) để xây dựng mô hình dự đoán hoạt tính sinh học.
  + Chia dữ liệu thành tập huấn luyện, tập kiểm tra và tập xác thực.
  + Sử dụng các metrics phù hợp (ví dụ: RMSE, R-squared, AUC) để đánh giá hiệu suất của mô hình.
* **Interpretation & Validation (Diễn giải và xác thực):**
  + Diễn giải các kết quả của mô hình để hiểu rõ hơn về mối quan hệ giữa cấu trúc và hoạt tính.
  + Xác thực mô hình bằng cách sử dụng dữ liệu bên ngoài hoặc thực hiện các thí nghiệm sinh học.

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

Dưới đây là hướng dẫn chi tiết bằng cả tiếng Anh và tiếng Việt cho từng bước trong quy trình phân tích:

**a. Kết nối cơ sở dữ liệu (Database Connection)**

**Tiếng Anh:**

import psycopg2  
  
# Database credentials  
db\_params = {  
 'host': '192.168.206.136',  
 'user': 'rd',  
 'password': 'rd',  
 'database': 'chembl\_35'  
}  
  
# Function to connect to the database  
def connect\_to\_db(params):  
 try:  
 conn = psycopg2.connect(\*\*params)  
 return conn  
 except psycopg2.Error as e:  
 print(f"Error connecting to the database: {e}")  
 return None  
  
conn = connect\_to\_db(db\_params)  
if conn:  
 print("Successfully connected to the database!")  
 conn.close()

**Tiếng Việt:**

import psycopg2  
  
# Thông tin đăng nhập cơ sở dữ liệu  
db\_params = {  
 'host': '192.168.206.136',  
 'user': 'rd',  
 'password': 'rd',  
 'database': 'chembl\_35'  
}  
  
# Hàm kết nối đến cơ sở dữ liệu  
def connect\_to\_db(params):  
 try:  
 conn = psycopg2.connect(\*\*params)  
 return conn  
 except psycopg2.Error as e:  
 print(f"Lỗi kết nối đến cơ sở dữ liệu: {e}")  
 return None  
  
conn = connect\_to\_db(db\_params)  
if conn:  
 print("Kết nối thành công đến cơ sở dữ liệu!")  
 conn.close()

**b. Truy vấn dữ liệu từ ChEMBL (Data Querying from ChEMBL)**

**Tiếng Anh:**

-- SQL query to retrieve 100 rows of activity data with specific criteria  
SELECT act.molregno, act.standard\_type, act.standard\_value, act.standard\_units,  
 md.chembl\_id, cs.canonical\_smiles  
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' -- Example activity type  
 AND act.standard\_units = 'nM' -- Example unit  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0 -- Ensure values are positive  
LIMIT 100;

**Tiếng Việt:**

-- Truy vấn SQL để lấy 100 dòng dữ liệu hoạt tính với các tiêu chí cụ thể  
SELECT act.molregno, act.standard\_type, act.standard\_value, act.standard\_units,  
 md.chembl\_id, cs.canonical\_smiles  
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' -- Ví dụ loại hoạt tính  
 AND act.standard\_units = 'nM' -- Ví dụ đơn vị đo  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0 -- Đảm bảo giá trị dương  
LIMIT 100;

**c. Đọc dữ liệu CSV và tiền xử lý (CSV Data Reading and Preprocessing)**

**Tiếng Anh:**

import pandas as pd  
import numpy as np  
  
# Define the base path for data  
base\_path = '../data'  
csv\_file\_path = os.path.join(base\_path, 'chembl\_activity\_data.csv')  
  
# Read the CSV file into a pandas DataFrame  
try:  
 df = pd.read\_csv(csv\_file\_path)  
except FileNotFoundError:  
 print(f"Error: The file {csv\_file\_path} was not found.")  
 exit()  
  
# Print the first few rows of the DataFrame  
print("First 5 rows of the dataframe:")  
print(df.head())  
  
# Basic data cleaning: drop rows with missing values in 'standard\_value' and 'canonical\_smiles'  
df.dropna(subset=['standard\_value', 'canonical\_smiles'], inplace=True)  
  
# Convert 'standard\_value' to numeric, errors='coerce' will turn non-convertible values into NaN  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce')  
  
# Remove rows where 'standard\_value' is NaN after conversion  
df.dropna(subset=['standard\_value'], inplace=True)  
  
# Optional: Convert IC50 to pIC50 (negative log scale)  
# Add a small constant to avoid log(0) errors  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Convert nM to M  
  
print("\nDataFrame info after cleaning:")  
print(df.info())  
  
print("\nSummary statistics for numeric columns:")  
print(df.describe())

**Tiếng Việt:**

import pandas as pd  
import numpy as np  
  
# Định nghĩa đường dẫn gốc cho dữ liệu  
base\_path = '../data'  
csv\_file\_path = os.path.join(base\_path, 'chembl\_activity\_data.csv')  
  
# Đọc file CSV vào DataFrame của pandas  
try:  
 df = pd.read\_csv(csv\_file\_path)  
except FileNotFoundError:  
 print(f"Lỗi: Không tìm thấy file {csv\_file\_path}.")  
 exit()  
  
# In ra vài dòng đầu của DataFrame  
print("5 dòng đầu của dataframe:")  
print(df.head())  
  
# Làm sạch dữ liệu cơ bản: loại bỏ các hàng có giá trị thiếu trong 'standard\_value' và 'canonical\_smiles'  
df.dropna(subset=['standard\_value', 'canonical\_smiles'], inplace=True)  
  
# Chuyển đổi 'standard\_value' sang kiểu số, errors='coerce' sẽ chuyển các giá trị không chuyển đổi được thành NaN  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce')  
  
# Loại bỏ các hàng có 'standard\_value' là NaN sau khi chuyển đổi  
df.dropna(subset=['standard\_value'], inplace=True)  
  
# Tùy chọn: Chuyển đổi IC50 thành pIC50 (thang log âm)  
# Thêm một hằng số nhỏ để tránh lỗi log(0)  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Chuyển đổi nM thành M  
  
print("\nThông tin DataFrame sau khi làm sạch:")  
print(df.info())  
  
print("\nThống kê mô tả cho các cột số:")  
print(df.describe())

**d. Tính toán đặc trưng hóa học với RDKit (Chemical Feature Calculation with RDKit)**

**Tiếng Anh:**

from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.Chem import Descriptors  
  
# Function to calculate RDKit descriptors  
def calculate\_descriptors(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 descriptors = {}  
 descriptors['MW'] = Descriptors.MolWt(mol)  
 descriptors['LogP'] = Descriptors.MolLogP(mol)  
 descriptors['HBA'] = Descriptors.NumHAcceptors(mol)  
 descriptors['HBD'] = Descriptors.NumHDonors(mol)  
 return descriptors  
 except Exception as e:  
 print(f"Error calculating descriptors: {e}")  
 return None  
  
# Apply the function to each row of the DataFrame  
df['descriptors'] = df['canonical\_smiles'].apply(calculate\_descriptors)  
  
# Convert the 'descriptors' column (which contains dictionaries) into separate columns  
df = pd.concat([df, df['descriptors'].apply(pd.Series)], axis=1)  
  
# Drop the original 'descriptors' column  
df.drop(columns=['descriptors'], inplace=True)  
  
# Print the first few rows with the new descriptor columns  
print(df.head())

**Tiếng Việt:**

from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.Chem import Descriptors  
  
# Hàm tính toán các descriptor RDKit  
def calculate\_descriptors(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 descriptors = {}  
 descriptors['MW'] = Descriptors.MolWt(mol)  
 descriptors['LogP'] = Descriptors.MolLogP(mol)  
 descriptors['HBA'] = Descriptors.NumHAcceptors(mol)  
 descriptors['HBD'] = Descriptors.NumHDonors(mol)  
 return descriptors  
 except Exception as e:  
 print(f"Lỗi khi tính toán descriptors: {e}")  
 return None  
  
# Áp dụng hàm cho mỗi hàng của DataFrame  
df['descriptors'] = df['canonical\_smiles'].apply(calculate\_descriptors)  
  
# Chuyển đổi cột 'descriptors' (chứa các dictionary) thành các cột riêng biệt  
df = pd.concat([df, df['descriptors'].apply(pd.Series)], axis=1)  
  
# Xóa cột 'descriptors' gốc  
df.drop(columns=['descriptors'], inplace=True)  
  
# In ra vài dòng đầu với các cột descriptor mới  
print(df.head())

**3. Ví dụ code SQL và Python (SQL and Python Code Examples)**

Dưới đây là 5 ví dụ code SQL và Python mẫu bạn có thể sử dụng làm điểm khởi đầu:

**Ví dụ 1: Truy vấn dữ liệu hoạt tính cho một mục tiêu cụ thể (Query activity data for a specific target)**

**SQL:**

SELECT act.molregno, act.standard\_type, act.standard\_value, md.chembl\_id  
FROM activities act  
JOIN molecule\_dictionary md ON act.molregno = md.molregno  
WHERE act.standard\_type = 'IC50'  
 AND act.target\_chembl\_id = 'CHEMBL205' -- Ví dụ: EGFR  
LIMIT 100;

**Python:**

import pandas as pd  
import psycopg2  
  
# Database credentials  
db\_params = {  
 'host': '192.168.206.136',  
 'user': 'rd',  
 'password': 'rd',  
 'database': 'chembl\_35'  
}  
  
# Function to connect to the database  
def connect\_to\_db(params):  
 try:  
 conn = psycopg2.connect(\*\*params)  
 return conn  
 except psycopg2.Error as e:  
 print(f"Error connecting to the database: {e}")  
 return None  
  
conn = connect\_to\_db(db\_params)  
if conn:  
 # SQL query to retrieve data for a specific target  
 sql\_query = """  
 SELECT act.molregno, act.standard\_type, act.standard\_value, md.chembl\_id  
 FROM activities act  
 JOIN molecule\_dictionary md ON act.molregno = md.molregno  
 WHERE act.standard\_type = 'IC50'  
 AND act.target\_chembl\_id = 'CHEMBL205'  
 LIMIT 100;  
 """  
  
 try:  
 # Use pandas to execute the SQL query and load the data into a DataFrame  
 df = pd.read\_sql\_query(sql\_query, conn)  
 print("Data loaded into DataFrame:")  
 print(df.head())  
 except psycopg2.Error as e:  
 print(f"Error executing SQL query: {e}")  
 finally:  
 conn.close() # Ensure the connection is closed in all cases  
else:  
 print("Failed to connect to the database.")

**Ví dụ 2: Tính toán fingerprints ECFP4 (Calculate ECFP4 fingerprints)**

**Python:**

from rdkit import Chem  
from rdkit.Chem import AllChem  
import pandas as pd  
  
# Assuming you have a DataFrame named 'df' with a 'canonical\_smiles' column  
def calculate\_ecfp4(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)  
 return fp.ToBitString() # Convert to bit string for easier handling  
 else:  
 return None  
 except Exception as e:  
 print(f"Error calculating ECFP4 fingerprint: {e}")  
 return None  
  
# Apply the function to each row of the DataFrame  
df['ECFP4'] = df['canonical\_smiles'].apply(calculate\_ecfp4)  
  
# Remove rows where ECFP4 is None  
df.dropna(subset=['ECFP4'], inplace=True)  
  
# Print the first few rows with the new ECFP4 column  
print(df.head())

**Ví dụ 3: Phân tích phân phối IC50 (IC50 distribution analysis)**

**Python:**

import matplotlib.pyplot as plt  
import seaborn as sns  
import pandas as pd  
  
# Assuming you have a DataFrame named 'df' with a 'standard\_value' column  
# Data Cleaning (Remove rows where 'standard\_value' is NaN)  
df = df.dropna(subset=['standard\_value'])  
  
# Convert 'standard\_value' to numeric type  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce')  
  
# Drop rows with NaN values that might have resulted from conversion  
df = df.dropna(subset=['standard\_value'])  
  
# Set up the matplotlib figure  
plt.figure(figsize=(10, 6))  
  
# Histogram  
sns.histplot(df['standard\_value'], kde=True, color='skyblue')  
plt.title('Distribution of IC50 Values')  
plt.xlabel('IC50 (nM)')  
plt.ylabel('Frequency')  
plt.show()  
  
# Boxplot  
plt.figure(figsize=(8, 6))  
sns.boxplot(x=df['standard\_value'], color='salmon')  
plt.title('Boxplot of IC50 Values')  
plt.xlabel('IC50 (nM)')  
plt.show()

**Ví dụ 4: Xây dựng mô hình hồi quy tuyến tính (Build a linear regression model)**

**Python:**

from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
import pandas as pd  
  
# Assuming you have a DataFrame named 'df' with descriptor columns (e.g., 'MW', 'LogP', 'HBA', 'HBD') and a 'pIC50' column  
# First, handle missing or infinite values. Replace them with NaN:  
df.replace([float('inf'), -float('inf')], float('nan'), inplace=True)  
  
# Drop rows with NaN values:  
df.dropna(inplace=True)  
  
# Define features (X) and target (y)  
X = df[['MW', 'LogP', 'HBA', 'HBD']]  
y = df['pIC50']  
  
# 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)  
  
# Create a linear regression model  
model = LinearRegression()  
  
# Train the model  
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}")

**Ví dụ 5: Phân tích tương quan giữa các đặc trưng (Correlation analysis between features)**

**Python:**

import matplotlib.pyplot as plt  
import seaborn as sns  
import pandas as pd  
  
# Assuming you have a DataFrame named 'df' with descriptor columns (e.g., 'MW', 'LogP', 'HBA', 'HBD', 'pIC50')  
# Correlation matrix  
corr\_matrix = df[['MW', 'LogP', 'HBA', 'HBD', 'pIC50']].corr()  
  
# Plot the correlation matrix using seaborn  
plt.figure(figsize=(10, 8))  
sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', linewidths=.5)  
plt.title('Correlation Matrix of Molecular Descriptors and pIC50')  
plt.show()

**Lưu ý về lỗi (Error Notes)**

* **Lỗi ERROR: operator does not exist: numeric ~ unknown:** Lỗi này xảy ra khi bạn sử dụng toán tử ~ (regular expression match) trên một cột số. Để khắc phục, bạn có thể ép kiểu cột standard\_value sang kiểu text trước khi so sánh, hoặc sử dụng các hàm số học để so sánh giá trị. Ví dụ:

-- Thay thế bằng cách ép kiểu sang text  
AND CAST(act.standard\_value AS TEXT) ~ '^[0-9\.]+$'  
  
-- Hoặc sử dụng điều kiện số học  
AND act.standard\_value >= 0

* **Lỗi 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. Bạn có thể tính căn bậc hai của MSE để có được RMSE:

from sklearn.metrics import mean\_squared\_error  
import numpy as np  
  
mse = mean\_squared\_error(y\_test, y\_pred)  
rmse = np.sqrt(mse)  
print(f"Root Mean Squared Error: {rmse}")

**Lưu ý quan trọng:**

* Hãy nhớ điều chỉnh các truy vấn SQL và code Python cho phù hợp với cấu trúc dữ liệu cụ thể và mục tiêu phân tích của bạn.
* Sử dụng os.path.join(base\_path, ...) để đảm bảo tính di động của code khi làm việc với đường dẫn file.
* Chia nhỏ các bước phân tích và kiểm tra kết quả thường xuyên để dễ dàng phát hiện và sửa lỗi.

Chúc bạn thành công với dự án phân tích dữ liệu ChEMBL 35! Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi.