# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Biểu đồ nhiệt về số lượng ligand tương tác 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 ChEMBL để tìm kiếm các hợp chất tiềm năng. Chúng ta sẽ cùng nhau xây dựng quy trình phân tích hiệu quả, đảm bảo tính chính xác và khả năng ứng dụng cao.

Dưới đây là phân tích chi tiết và hướng dẫn song ngữ, cùng với code SQL và Python mẫu, được thiết kế để giải quyết các vấn đề bạn đã nêu và tuân thủ theo quy trình làm việc của bạn:

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

Mục tiêu của chúng ta là khai thác dữ liệu từ ChEMBL 35 để hỗ trợ nghiên cứu và phát triển thuốc. Dưới đây là mô hình phân tích đề xuất:

* **Data Extraction (Trích xuất dữ liệu):** Sử dụng SQL để truy vấn và trích xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL 35 (sử dụng PostgreSQL). Dữ liệu này có thể bao gồm thông tin về cấu trúc hóa học, hoạt tính sinh học, thuộc tính dược động học, v.v.
* **Data Preprocessing (Tiền xử lý dữ liệu):** Sử dụng RDKit và Python để làm sạch, chuyển đổi và chuẩn hóa dữ liệu. Các bước có thể bao gồm:
  + Chuyển đổi SMILES sang các định dạng cấu trúc khác (ví dụ: Mol).
  + Tính toán các descriptor phân tử (ví dụ: MW, LogP, TPSA).
  + Xử lý các giá trị bị thiếu hoặc không hợp lệ.
* **Exploratory Data Analysis (EDA) (Phân tích thăm dò dữ liệu):** Sử dụng Python (với các thư viện như Pandas, Matplotlib, Seaborn) để khám phá dữ liệu, tìm kiếm các mẫu và mối quan hệ tiềm năng.
* **Model Building (Xây dựng mô hình):** Xây dựng các mô hình dự đoán (ví dụ: mô hình QSAR/QSPR) để dự đoán hoạt tính sinh học hoặc các thuộc tính khác của các hợp chất dựa trên cấu trúc của chúng.
* **Model Validation (Xác thực mô hình):** Đánh giá hiệu suất của mô hình bằng cách sử dụng các tập dữ liệu kiểm tra và các chỉ số phù hợp.
* **Interpretation and Application (Giải thích và ứng dụng):** Giải thích kết quả của mô hình và sử dụng chúng để đưa ra các quyết định sáng suốt trong quá trình phát triển thuốc.

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

Dưới đây là hướng dẫn chi tiết cho từng bước trong quy trình phân tích, kèm theo ví dụ code SQL và Python.

**Bước 1: Trích xuất dữ liệu từ ChEMBL 35 (Data Extraction from ChEMBL 35)**

* **SQL (English):**

-- Select compound properties and activities for a specific target  
SELECT  
 cmp.chembl\_id,  
 cmp.canonical\_smiles,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 compound\_structures cmp  
JOIN  
 activities act ON cmp.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.target\_name = 'CHEMBL35 Target Name' -- Replace with the actual target name  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Only numeric values  
LIMIT 100;

* **SQL (Tiếng Việt):**

-- Lựa chọn thuộc tính hợp chất và hoạt tính cho một mục tiêu cụ thể  
SELECT  
 cmp.chembl\_id,  
 cmp.canonical\_smiles,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 compound\_structures cmp  
JOIN  
 activities act ON cmp.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.target\_name = 'Tên mục tiêu CHEMBL35' -- Thay thế bằng tên mục tiêu thực tế  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Chỉ các giá trị số  
LIMIT 100;

**Lưu ý:**

* Sửa lỗi ERROR: operator does not exist: numeric ~ unknown: Đảm bảo cột act.standard\_value là kiểu số (numeric). Nếu không, bạn cần chuyển đổi nó trước khi so sánh. Tuy nhiên, việc sử dụng ~ '^[0-9\.]+$' để lọc giá trị số là không cần thiết nếu cột đã là kiểu số. Bạn chỉ cần đảm bảo nó không null.
* Thay thế 'CHEMBL35 Target Name' bằng tên mục tiêu cụ thể mà bạn quan tâm.
* LIMIT 100 để chỉ lấy 100 dòng dữ liệu đầu tiên.

**Bước 2: Tiền xử lý dữ liệu bằng RDKit và Python (Data Preprocessing with RDKit and Python)**

* **Python (English):**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import os  
  
# Define the base path  
base\_path = "../data" # Adjust as needed  
  
# Load data from CSV  
csv\_file\_path = os.path.join(base\_path, "your\_data.csv") # Replace 'your\_data.csv'  
df = pd.read\_csv(csv\_file\_path)  
  
# Function to calculate molecular weight  
def calculate\_mw(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Descriptors.MolWt(mol)  
 else:  
 return None  
  
# Apply the function to the 'canonical\_smiles' column  
df['molecular\_weight'] = df['canonical\_smiles'].apply(calculate\_mw)  
  
# Handle missing values (example: fill with the mean)  
df['molecular\_weight'] = df['molecular\_weight'].fillna(df['molecular\_weight'].mean())  
  
print(df.head())

* **Python (Tiếng Việt):**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import os  
  
# Định nghĩa đường dẫn gốc  
base\_path = "../data" # Điều chỉnh nếu cần  
  
# Tải dữ liệu từ file CSV  
csv\_file\_path = os.path.join(base\_path, "your\_data.csv") # Thay thế 'your\_data.csv'  
df = pd.read\_csv(csv\_file\_path)  
  
# Hàm tính toán trọng lượng phân tử  
def calculate\_mw(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Descriptors.MolWt(mol)  
 else:  
 return None  
  
# Áp dụng hàm cho cột 'canonical\_smiles'  
df['molecular\_weight'] = df['canonical\_smiles'].apply(calculate\_mw)  
  
# Xử lý giá trị thiếu (ví dụ: điền bằng giá trị trung bình)  
df['molecular\_weight'] = df['molecular\_weight'].fillna(df['molecular\_weight'].mean())  
  
print(df.head())

**Lưu ý:**

* Thay thế "your\_data.csv" bằng tên file CSV thực tế của bạn.
* Điều chỉnh base\_path nếu cần thiết.
* Đảm bảo bạn đã cài đặt RDKit: conda install -c conda-forge rdkit

**Bước 3: Phân tích thăm dò dữ liệu (Exploratory Data Analysis - EDA)**

* **Python (English):**

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
import os  
  
# Define the base path  
base\_path = "../data" # Adjust as needed  
  
# Load data from CSV  
csv\_file\_path = os.path.join(base\_path, "your\_data.csv") # Replace 'your\_data.csv'  
df = pd.read\_csv(csv\_file\_path)  
  
# Basic statistics  
print(df.describe())  
  
# Distribution of 'standard\_value'  
sns.histplot(df['standard\_value'])  
plt.title('Distribution of Standard Value')  
plt.show()  
  
# Scatter plot of 'molecular\_weight' vs. 'standard\_value'  
sns.scatterplot(x='molecular\_weight', y='standard\_value', data=df)  
plt.title('Molecular Weight vs. Standard Value')  
plt.show()

* **Python (Tiếng Việt):**

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
import os  
  
# Định nghĩa đường dẫn gốc  
base\_path = "../data" # Điều chỉnh nếu cần  
  
# Tải dữ liệu từ file CSV  
csv\_file\_path = os.path.join(base\_path, "your\_data.csv") # Thay thế 'your\_data.csv'  
df = pd.read\_csv(csv\_file\_path)  
  
# Thống kê cơ bản  
print(df.describe())  
  
# Phân phối của 'standard\_value'  
sns.histplot(df['standard\_value'])  
plt.title('Phân phối của Standard Value')  
plt.show()  
  
# Biểu đồ phân tán của 'molecular\_weight' so với 'standard\_value'  
sns.scatterplot(x='molecular\_weight', y='standard\_value', data=df)  
plt.title('Trọng lượng phân tử so với Standard Value')  
plt.show()

**Bước 4: Xây dựng và xác thực mô hình (Model Building and Validation - ví dụ đơn giản)**

* **Python (English):**

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
  
# Load data from CSV  
df = pd.read\_csv("your\_data.csv") # Replace 'your\_data.csv'  
  
# Assuming 'molecular\_weight' is your feature and 'standard\_value' is your target  
X = df[['molecular\_weight']].dropna()  
y = df['standard\_value'].dropna()  
  
# 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  
y\_pred = model.predict(X\_test)  
  
# Evaluate the model  
mse = mean\_squared\_error(y\_test, y\_pred, squared=False) # squared=False for RMSE  
print(f"Root Mean Squared Error: {mse}")

* **Python (Tiếng Việt):**

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
  
# Tải dữ liệu từ file CSV  
df = pd.read\_csv("your\_data.csv") # Thay thế 'your\_data.csv'  
  
# Giả sử 'molecular\_weight' là đặc trưng và 'standard\_value' là mục tiêu  
X = df[['molecular\_weight']].dropna()  
y = df['standard\_value'].dropna()  
  
# 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)  
  
# Tạo mô hình hồi quy tuyến tính  
model = LinearRegression()  
  
# Huấn luyện mô hình  
model.fit(X\_train, y\_train)  
  
# Dự đoán  
y\_pred = model.predict(X\_test)  
  
# Đánh giá mô hình  
mse = mean\_squared\_error(y\_test, y\_pred, squared=False) # squared=False cho RMSE  
print(f"Sai số bình phương gốc trung bình: {mse}")

**Lưu ý:**

* Sửa lỗi squared=False: Nếu phiên bản scikit-learn của bạn quá cũ, hãy nâng cấp nó: pip install -U scikit-learn. Nếu không thể nâng cấp, hãy tính căn bậc hai của MSE: mse = mean\_squared\_error(y\_test, y\_pred); rmse = mse\*\*0.5.
* Đây chỉ là một ví dụ đơn giản. Bạn có thể sử dụng các mô hình phức tạp hơn và nhiều đặc trưng hơn.
* Cần thực hiện cross-validation để đánh giá mô hình một cách chính xác.

**3. 5 Ví dụ Code SQL và Python Mẫu (5 Sample SQL and Python Code Examples)**

Dưới đây là 5 ví dụ bổ sung để minh họa các thao tác khác nhau:

**Ví dụ 1: Tính LogP sử dụng RDKit (Calculate LogP using RDKit)**

* **Python (English):**

from rdkit import Chem  
from rdkit.Chem import Crippen  
import pandas as pd  
  
def calculate\_logp(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Crippen.MolLogP(mol)  
 else:  
 return None  
  
# Example usage with a DataFrame  
df = pd.DataFrame({'smiles': ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1']}) #aspirin, benzene  
df['logp'] = df['smiles'].apply(calculate\_logp)  
print(df)

* **Python (Tiếng Việt):**

from rdkit import Chem  
from rdkit.Chem import Crippen  
import pandas as pd  
  
def tinh\_logp(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Crippen.MolLogP(mol)  
 else:  
 return None  
  
# Ví dụ sử dụng với DataFrame  
df = pd.DataFrame({'smiles': ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1']}) #aspirin, benzene  
df['logp'] = df['smiles'].apply(tinh\_logp)  
print(df)

**Ví dụ 2: Lọc các hợp chất có trọng lượng phân tử trong một khoảng nhất định (Filter compounds by molecular weight range)**

* **SQL (English):**

SELECT chembl\_id, canonical\_smiles  
FROM compound\_structures  
WHERE molregno IN (  
 SELECT molregno  
 FROM molecule\_dictionary  
 WHERE rlogp BETWEEN 2 AND 5 -- LogP between 2 and 5  
)  
LIMIT 100;

* **SQL (Tiếng Việt):**

SELECT chembl\_id, canonical\_smiles  
FROM compound\_structures  
WHERE molregno IN (  
 SELECT molregno  
 FROM molecule\_dictionary  
 WHERE rlogp BETWEEN 2 AND 5 -- LogP nằm giữa 2 và 5  
)  
LIMIT 100;

**Ví dụ 3: Tính TPSA (Topological Polar Surface Area) (Calculate TPSA)**

* **Python (English):**

from rdkit import Chem  
from rdkit.Chem import Descriptors  
import pandas as pd  
  
def calculate\_tpsa(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Descriptors.TPSA(mol)  
 else:  
 return None  
  
# Example usage  
df = pd.DataFrame({'smiles': ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1']})  
df['tpsa'] = df['smiles'].apply(calculate\_tpsa)  
print(df)

* **Python (Tiếng Việt):**

from rdkit import Chem  
from rdkit.Chem import Descriptors  
import pandas as pd  
  
def tinh\_tpsa(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Descriptors.TPSA(mol)  
 else:  
 return None  
  
# Ví dụ sử dụng  
df = pd.DataFrame({'smiles': ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1']})  
df['tpsa'] = df['smiles'].apply(tinh\_tpsa)  
print(df)

**Ví dụ 4: Lấy các hợp chất hoạt động trên một mục tiêu cụ thể (Get active compounds for a specific target)**

* **SQL (English):**

SELECT cmp.chembl\_id, cmp.canonical\_smiles, act.standard\_value  
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 the target ChEMBL ID  
AND act.standard\_type = 'IC50'  
AND act.standard\_value < 100 -- IC50 < 100 nM  
LIMIT 100;

* **SQL (Tiếng Việt):**

SELECT cmp.chembl\_id, cmp.canonical\_smiles, act.standard\_value  
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' -- Thay thế bằng ChEMBL ID của mục tiêu  
AND act.standard\_type = 'IC50'  
AND act.standard\_value < 100 -- IC50 < 100 nM  
LIMIT 100;

**Ví dụ 5: Chuyển đổi IC50 sang pIC50 (Convert IC50 to pIC50)**

* **Python (English):**

import numpy as np  
import pandas as pd  
  
def ic50\_to\_pic50(ic50\_nM):  
 """Converts IC50 (nM) to pIC50."""  
 pIC50 = -np.log10(ic50\_nM \* 1e-9) # Convert nM to Molar  
 return pIC50  
  
# Example Usage (assuming you have a DataFrame 'df' with an 'standard\_value' column representing IC50 in nM)  
df = pd.DataFrame({'standard\_value': [10, 100, 1000]})  
df['pIC50'] = df['standard\_value'].apply(ic50\_to\_pic50)  
print(df)

* **Python (Tiếng Việt):**

import numpy as np  
import pandas as pd  
  
def ic50\_sang\_pic50(ic50\_nM):  
 """Chuyển đổi IC50 (nM) thành pIC50."""  
 pIC50 = -np.log10(ic50\_nM \* 1e-9) # Chuyển đổi nM thành Molar  
 return pIC50  
  
# Ví dụ sử dụng (giả sử bạn có DataFrame 'df' với cột 'standard\_value' đại diện cho IC50 tính bằng nM)  
df = pd.DataFrame({'standard\_value': [10, 100, 1000]})  
df['pIC50'] = df['standard\_value'].apply(ic50\_sang\_pic50)  
print(df)

**Kết luận**

Hy vọng rằng phân tích và hướng dẫn chi tiết này sẽ giúp bạn tiến xa hơn trong dự án của mình. Đừng ngần ngại hỏi nếu bạn có bất kỳ câu hỏi nào hoặc cần hỗ trợ thêm. Chúc bạn thành công!