# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Biểu đồ so sánh các lớp hợp chất

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit này. Với vai trò là một chuyên gia nghiên cứu và phát triển dược học, tôi hiểu 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, tập trung vào mã chủ đề “Topic\_CheMBL\_35\_75”.

**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, sử dụng RDKit để tính toán các đặc tính lý hóa.

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

Mục tiêu của chủ đề này là khám phá mối liên hệ giữa cấu trúc hóa học (được biểu diễn thông qua các đặc tính lý hóa) và hoạt tính sinh học của các hợp chất. Mô hình phân tích sẽ bao gồm các bước sau:

* **Trích xuất dữ liệu từ ChEMBL 35:** Sử dụng SQL để truy vấn và lấy dữ liệu về các hợp chất (molecules) và hoạt tính sinh học của chúng (activities).
* **Tính toán đặc tính lý hóa bằng RDKit:** Sử dụng thư viện RDKit để tính toán các đặc tính như trọng lượng phân tử (molecular weight), LogP, số lượng liên kết hydro cho (hydrogen bond donors), số lượng liên kết hydro nhận (hydrogen bond acceptors), diện tích bề mặt phân cực (polar surface area),…
* **Tiền xử lý dữ liệu:** Làm sạch và chuẩn hóa dữ liệu, xử lý các giá trị thiếu (missing values) và ngoại lệ (outliers).
* **Phân tích tương quan:** Sử dụng các phương pháp thống kê (như hệ số tương quan Pearson, Spearman) và/hoặc các mô hình học máy (machine learning models) để đánh giá mối tương quan giữa các đặc tính lý hóa và hoạt tính sinh học. Các mô hình học máy có thể bao gồm hồi quy tuyến tính (linear regression), cây quyết định (decision trees), random forests, và mạng nơ-ron (neural networks).
* **Trực quan hóa dữ liệu:** Sử dụng các biểu đồ (scatter plots, histograms, box plots) để trực quan hóa các mối tương quan và kết quả phân tích.

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

**2.1. SQL (English & Vietnamese):**

* **English:** SQL will be used to extract data from the ChEMBL 35 database, including molecule information and their corresponding bioactivities.
* **Vietnamese:** SQL sẽ được sử dụng để trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL 35, bao gồm thông tin về các phân tử và hoạt tính sinh học tương ứng của chúng.

**2.2. Python (English & Vietnamese):**

* **English:** Python, with libraries like RDKit, Pandas, NumPy, and Scikit-learn, will be used for data manipulation, physicochemical property calculation, statistical analysis, and machine learning model building.
* **Vietnamese:** Python, với các thư viện như RDKit, Pandas, NumPy và Scikit-learn, sẽ được sử dụng để xử lý dữ liệu, tính toán các đặc tính lý hóa, phân tích thống kê và xây dựng mô hình học máy.

**3. Code mẫu (Code Examples):**

**3.1. SQL (chiết xuất dữ liệu, giới hạn 100 dòng):**

-- English  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
WHERE  
 act.standard\_type = 'IC50' -- You can change this to other activity types  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0 -- Filter out invalid activity values  
 AND act.standard\_value ~ '^[0-9\.]+$' -- added this line to solve the error  
LIMIT 100;  
  
-- Vietnamese  
-- Lấy thông tin về phân tử, cấu trúc SMILES, loại hoạt tính, giá trị và đơn vị  
-- từ các bảng molecule\_dictionary, compound\_structures và activities.  
-- Lọc theo loại hoạt tính là IC50, đơn vị là nM, và giới hạn kết quả 100 dòng.  
SELECT  
 md.molregno, -- Mã số phân tử  
 cs.canonical\_smiles, -- Cấu trúc SMILES  
 act.standard\_type, -- Loại hoạt tính (ví dụ: IC50)  
 act.standard\_value, -- Giá trị hoạt tính  
 act.standard\_units -- Đơn vị hoạt tính  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
WHERE  
 act.standard\_type = 'IC50' -- Lọc theo loại hoạt tính IC50  
 AND act.standard\_units = 'nM' -- Lọc theo đơn vị nM  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0 -- Loại bỏ giá trị hoạt tính không hợp lệ  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Added this line to solve the error  
LIMIT 100; -- Giới hạn kết quả 100 dòng

**Lưu ý sửa lỗi SQL:**

Lỗi ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard\_value ~ '^[0-9\.]+$' xảy ra vì bạn đang cố gắng so sánh một cột kiểu số (numeric) với một biểu thức chính quy (regular expression). Để giải quyết vấn đề này, bạn cần đảm bảo rằng cột act.standard\_value chỉ chứa các giá trị số (và dấu chấm thập phân). Bạn có thể thêm điều kiện AND act.standard\_value ~ '^[0-9\.]+$' để lọc ra các giá trị không hợp lệ. Hoặc bạn có thể ép kiểu dữ liệu (cast) cột act.standard\_value sang kiểu text trước khi so sánh.

**3.2. Python (tính toán LogP bằng RDKit):**

# English  
import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.Chem import Descriptors  
  
base\_path = "." # Assuming the notebook is in the root of your project  
data\_path = os.path.join(base\_path, "data")  
  
# Assuming you have a CSV file named "chembl\_data.csv" in the 'data' folder  
data\_file = os.path.join(data\_path, "chembl\_data.csv")  
  
# Create the 'data' directory if it doesn't exist  
if not os.path.exists(data\_path):  
 os.makedirs(data\_path)  
  
# Check if the data file exists  
if not os.path.exists(data\_file):  
 print(f"Error: The data file '{data\_file}' does not exist. Please make sure you have the file in the correct location.")  
else:  
 df = pd.read\_csv(data\_file)  
  
 # Function to calculate LogP using RDKit  
 def calculate\_logp(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 logp = Descriptors.MolLogP(mol)  
 return logp  
 else:  
 return None  
 except:  
 return None  
  
 # Apply the function to the 'canonical\_smiles' column  
 df['logp'] = df['canonical\_smiles'].apply(calculate\_logp)  
  
 # Print the first 10 rows with LogP values  
 print(df[['canonical\_smiles', 'logp']].head(10))  
  
# Vietnamese  
import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.Chem import Descriptors  
  
base\_path = "." # Giả sử notebook nằm ở thư mục gốc của dự án  
data\_path = os.path.join(base\_path, "data")  
  
# Giả sử bạn có một file CSV tên là "chembl\_data.csv" trong thư mục 'data'  
data\_file = os.path.join(data\_path, "chembl\_data.csv")  
  
# Tạo thư mục 'data' nếu nó chưa tồn tại  
if not os.path.exists(data\_path):  
 os.makedirs(data\_path)  
  
# Kiểm tra xem file dữ liệu có tồn tại không  
if not os.path.exists(data\_file):  
 print(f"Lỗi: File dữ liệu '{data\_file}' không tồn tại. Vui lòng đảm bảo bạn đã đặt file đúng vị trí.")  
else:  
 df = pd.read\_csv(data\_file)  
  
 # Hàm tính toán LogP sử dụng RDKit  
 def calculate\_logp(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 logp = Descriptors.MolLogP(mol)  
 return logp  
 else:  
 return None  
 except:  
 return None  
  
 # Áp dụng hàm cho cột 'canonical\_smiles'  
 df['logp'] = df['canonical\_smiles'].apply(calculate\_logp)  
  
 # In ra 10 dòng đầu tiên với giá trị LogP  
 print(df[['canonical\_smiles', 'logp']].head(10))

**3.3. Python (phân tích tương quan giữa LogP và pIC50):**

# English  
import os  
import pandas as pd  
import numpy as np  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from scipy.stats import pearsonr  
from sklearn.metrics import mean\_squared\_error  
  
base\_path = "." # Assuming the notebook is in the root of your project  
data\_path = os.path.join(base\_path, "data")  
data\_file = os.path.join(data\_path, "chembl\_data.csv")  
  
if not os.path.exists(data\_file):  
 print(f"Error: The data file '{data\_file}' does not exist. Please make sure you have the file in the correct location.")  
else:  
 df = pd.read\_csv(data\_file)  
  
 # Function to calculate LogP  
 def calculate\_logp(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 logp = Descriptors.MolLogP(mol)  
 return logp  
 else:  
 return None  
 except:  
 return None  
  
 df['logp'] = df['canonical\_smiles'].apply(calculate\_logp)  
  
 # Convert IC50 to pIC50  
 df['pIC50'] = -np.log10(df['standard\_value'] / 1e9) # Convert nM to M  
  
 # Drop rows with missing LogP or pIC50 values  
 df = df.dropna(subset=['logp', 'pIC50'])  
  
 # Calculate Pearson correlation coefficient  
 corr, p\_value = pearsonr(df['logp'], df['pIC50'])  
 print(f"Pearson correlation between LogP and pIC50: {corr:.2f} (p-value: {p\_value:.3f})")  
  
 # Simple Linear Regression (as an example, not necessarily the best model)  
 from sklearn.linear\_model import LinearRegression  
 from sklearn.model\_selection import train\_test\_split  
  
 X = df[['logp']]  
 y = df['pIC50']  
  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
 model = LinearRegression()  
 model.fit(X\_train, y\_train)  
  
 y\_pred = model.predict(X\_test)  
  
 # Calculate Mean Squared Error  
 mse = mean\_squared\_error(y\_test, y\_pred)  
 print(f"Mean Squared Error: {mse:.2f}")  
  
  
# Vietnamese  
import os  
import pandas as pd  
import numpy as np  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from scipy.stats import pearsonr  
from sklearn.metrics import mean\_squared\_error  
  
base\_path = "." # Giả sử notebook nằm ở thư mục gốc của dự án  
data\_path = os.path.join(base\_path, "data")  
data\_file = os.path.join(data\_path, "chembl\_data.csv")  
  
if not os.path.exists(data\_file):  
 print(f"Lỗi: File dữ liệu '{data\_file}' không tồn tại. Vui lòng đảm bảo bạn đã đặt file đúng vị trí.")  
else:  
 df = pd.read\_csv(data\_file)  
  
 # Hàm tính toán LogP  
 def calculate\_logp(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 logp = Descriptors.MolLogP(mol)  
 return logp  
 else:  
 return None  
 except:  
 return None  
  
 df['logp'] = df['canonical\_smiles'].apply(calculate\_logp)  
  
 # Chuyển đổi IC50 sang pIC50  
 df['pIC50'] = -np.log10(df['standard\_value'] / 1e9) # Chuyển đổi nM sang M  
  
 # Loại bỏ các hàng có giá trị LogP hoặc pIC50 bị thiếu  
 df = df.dropna(subset=['logp', 'pIC50'])  
  
 # Tính toán hệ số tương quan Pearson  
 corr, p\_value = pearsonr(df['logp'], df['pIC50'])  
 print(f"Tương quan Pearson giữa LogP và pIC50: {corr:.2f} (p-value: {p\_value:.3f})")  
  
 # Hồi quy tuyến tính đơn giản (ví dụ, không nhất thiết là mô hình tốt nhất)  
 from sklearn.linear\_model import LinearRegression  
 from sklearn.model\_selection import train\_test\_split  
  
 X = df[['logp']]  
 y = df['pIC50']  
  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
 model = LinearRegression()  
 model.fit(X\_train, y\_train)  
  
 y\_pred = model.predict(X\_test)  
  
 # Tính toán sai số bình phương trung bình  
 mse = mean\_squared\_error(y\_test, y\_pred)  
 print(f"Sai số bình phương trung bình: {mse:.2f}")

**Lưu ý sửa lỗi Python:**

* **mean\_squared\_error(y\_test, y\_pred, squared=False):** Nếu bạn đang sử dụng phiên bản scikit-learn cũ, tham số squared=False có thể không được hỗ trợ. Trong trường hợp đó, hãy bỏ tham số này đi và tính căn bậc hai của kết quả nếu bạn muốn tính RMSE (Root Mean Squared Error).

**3.4. SQL (Extracting Specific Molecules with Molecular Weight):**

-- English  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
WHERE  
 act.standard\_type = 'Ki' -- Example: Select based on inhibition constant  
 AND act.standard\_units = 'nM'  
 AND md.mw\_freebase BETWEEN 200 AND 400 -- Filter by molecular weight range  
LIMIT 100;  
  
-- Vietnamese  
-- Lấy thông tin phân tử với trọng lượng phân tử nằm trong khoảng nhất định.  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
WHERE  
 act.standard\_type = 'Ki' -- Chọn dựa trên hằng số ức chế  
 AND act.standard\_units = 'nM'  
 AND md.mw\_freebase BETWEEN 200 AND 400 -- Lọc theo khoảng trọng lượng phân tử  
LIMIT 100;

**3.5. Python (Calculating TPSA):**

# English  
import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data")  
data\_file = os.path.join(data\_path, "chembl\_data.csv")  
  
if not os.path.exists(data\_file):  
 print(f"Error: The data file '{data\_file}' does not exist.")  
else:  
 df = pd.read\_csv(data\_file)  
  
 def calculate\_tpsa(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 tpsa = Descriptors.TPSA(mol)  
 return tpsa  
 else:  
 return None  
 except:  
 return None  
  
 df['tpsa'] = df['canonical\_smiles'].apply(calculate\_tpsa)  
 print(df[['canonical\_smiles', 'tpsa']].head())  
  
# Vietnamese  
# Tính toán diện tích bề mặt phân cực (TPSA).  
import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
base\_path = "."  
data\_path = os.path.join(base\_path, "data")  
data\_file = os.path.join(data\_path, "chembl\_data.csv")  
  
if not os.path.exists(data\_file):  
 print(f"Lỗi: File dữ liệu '{data\_file}' không tồn tại.")  
else:  
 df = pd.read\_csv(data\_file)  
  
 def calculate\_tpsa(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 tpsa = Descriptors.TPSA(mol)  
 return tpsa  
 else:  
 return None  
 except:  
 return None  
  
 df['tpsa'] = df['canonical\_smiles'].apply(calculate\_tpsa)  
 print(df[['canonical\_smiles', 'tpsa']].head())

**Lưu ý quan trọng:**

* **Đường dẫn file:** Hãy đảm bảo rằng đường dẫn đến file CSV của bạn (data\_file) là chính xác. Sử dụng os.path.join để đảm bảo tính tương thích trên các hệ điều hành khác nhau.
* **Xử lý lỗi:** Luôn luôn có các khối try...except để xử lý các lỗi có thể xảy ra trong quá trình tính toán các đặc tính lý hóa.
* **Chọn đặc tính lý hóa phù hợp:** Tùy thuộc vào mục tiêu nghiên cứu của bạn, hãy chọn các đặc tính lý hóa phù hợp để phân tích.
* **Tiền xử lý dữ liệu cẩn thận:** Việc làm sạch và chuẩn hóa dữ liệu là rất quan trọng để đảm bảo kết quả phân tích chính xác.
* **Sử dụng các mô hình học máy phù hợp:** Nếu bạn sử dụng các mô hình học máy, hãy chọn các mô hình phù hợp với loại dữ liệu và mục tiêu của bạn. Đánh giá hiệu suất của mô hình bằng các phương pháp phù hợp (ví dụ: cross-validation).

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