# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Trực quan hóa họ protein từ ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Với vai trò là một chuyên gia R&D dược học, tôi hiểu rõ tầm quan trọng của việc khai thác thông tin từ cơ sở dữ liệu lớn như ChEMBL để thúc đẩy quá trình phát triển thuốc.

Dựa trên thông tin bạn cung cấp, tôi sẽ giúp bạn xây dựng quy trình phân tích, khắc phục lỗi và cung cấp các ví dụ code mẫu theo chuẩn AIMLOps.

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

Mục tiêu của bạn là phân tích dữ liệu ChEMBL 35 để phục vụ nghiên cứu và phát triển thuốc. Dưới đây là một số mô hình phân tích phổ biến bạn có thể áp dụng:

* **Phân tích SAR (Structure-Activity Relationship):** Nghiên cứu mối quan hệ giữa cấu trúc hóa học của các phân tử và hoạt tính sinh học của chúng. Điều này giúp xác định các nhóm thế (functional groups) quan trọng cho hoạt tính.
* **Phân tích QSAR (Quantitative Structure-Activity Relationship):** Xây dựng các mô hình toán học để dự đoán hoạt tính sinh học dựa trên các thuộc tính hóa học tính toán được (ví dụ: trọng lượng phân tử, logP, số lượng liên kết hydro).
* **Phân tích đa dạng hóa học (Chemical Diversity Analysis):** Đánh giá sự đa dạng của các phân tử trong một tập hợp dữ liệu, giúp chọn ra các phân tử đại diện cho các vùng không gian hóa học khác nhau.
* **Phân tích sàng lọc ảo (Virtual Screening):** Sử dụng các mô hình QSAR hoặc docking để sàng lọc một lượng lớn các phân tử và xác định các ứng cử viên tiềm năng cho thử nghiệm sinh học.
* **Phân tích liên kết hóa học (Chemical Linkage Analysis):** Nghiên cứu các liên kết hóa học có trong các phân tử có hoạt tính, để tìm ra các mẫu cấu trúc chung (common structural patterns) liên quan đến hoạt tính.

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

Tôi sẽ cung cấp hướng dẫn bằng cả tiếng Anh và tiếng Việt để bạn dễ dàng theo dõi và thực hiện.

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

Tôi sẽ cung cấp các đoạn code SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35 và các đoạn code Python sử dụng RDKit để xử lý và phân tích dữ liệu.

**4. Ví dụ Code (Code Examples):**

Tôi sẽ cung cấp 5 ví dụ code SQL và Python mẫu để bạn bắt đầu.

**Khắc phục lỗi (Error Handling):**

* **Lỗi SQL (SQL Error):** “ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard\_value ~ ‘[[1]](#footnote-21)+$’”
  + **Giải thích (Explanation):** Lỗi này xảy ra do bạn đang cố gắng sử dụng toán tử ~ (regex match) trên một cột kiểu số (numeric). Toán tử này chỉ dùng cho kiểu dữ liệu văn bản (text).
  + **Khắc phục (Solution):** Bạn cần chuyển đổi cột standard\_value sang kiểu văn bản trước khi sử dụng toán tử ~.
  + **English:** This error occurs because you are trying to use the ~ (regex match) operator on a numeric column. This operator is only for text data types. You need to convert the standard\_value column to text before using the ~ operator.
* **Lỗi Python (Python Error):** “phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean\_squared\_error”
  + **Giải thích (Explanation):** Phiên bản scikit-learn bạn đang sử dụng quá cũ và không hỗ trợ tham số squared=False trong hàm mean\_squared\_error.
  + **Khắc phục (Solution):** Nâng cấp phiên bản scikit-learn của bạn lên phiên bản mới nhất bằng lệnh pip install --upgrade scikit-learn. Nếu không thể nâng cấp, bạn có thể tính căn bậc hai của MSE (Mean Squared Error) để có được RMSE (Root Mean Squared Error).
  + **English:** The version of scikit-learn you are using is too old and does not support the squared=False parameter in the mean\_squared\_error function. Upgrade your scikit-learn version to the latest version using pip install --upgrade scikit-learn. If you cannot upgrade, you can calculate the square root of MSE to get RMSE.

**Ví dụ Code (Code Examples):**

Dưới đây là 5 ví dụ code SQL và Python mẫu, tập trung vào phân tích SAR và QSAR:

**Ví dụ 1: Lấy dữ liệu hoạt tính và cấu trúc (Fetching Activity and Structure Data)**

* **SQL:**

-- Lấy 100 dòng dữ liệu hoạt tính kháng một target cụ thể (ví dụ: CHEMBL240)  
SELECT  
 md.chembl\_id,  
 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.assay\_id IN (SELECT assay\_id FROM assays WHERE target\_chembl\_id = 'CHEMBL240')  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND cs.canonical\_smiles IS NOT NULL  
LIMIT 100;

* **English:**

-- Fetch 100 rows of activity data against a specific target (e.g., CHEMBL240)  
SELECT  
 md.chembl\_id,  
 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.assay\_id IN (SELECT assay\_id FROM assays WHERE target\_chembl\_id = 'CHEMBL240')  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND cs.canonical\_smiles IS NOT NULL  
LIMIT 100;

* **Python:**

import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
base\_path = "." # Thay đổi nếu cần  
  
# Đường dẫn đến file CSV chứa dữ liệu lấy từ SQL  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
  
# Đọc dữ liệu từ file CSV  
df = pd.read\_csv(csv\_file\_path)  
  
# Hàm tính toán các thuộc tính hóa học sử dụng RDKit  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 descriptors = {desc\_name: desc\_func(mol) for desc\_name, desc\_func in Descriptors.descList}  
 return descriptors  
 else:  
 return None  
  
# Áp dụng hàm tính toán thuộc tính cho cột 'canonical\_smiles'  
df['descriptors'] = df['canonical\_smiles'].apply(calculate\_descriptors)  
  
# In ra một vài dòng đầu của DataFrame  
print(df.head())

* **English:**

import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
base\_path = "." # Change if needed  
  
# Path to the CSV file containing data fetched from SQL  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
  
# Read data from CSV file  
df = pd.read\_csv(csv\_file\_path)  
  
# Function to calculate chemical descriptors using RDKit  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 descriptors = {desc\_name: desc\_func(mol) for desc\_name, desc\_func in Descriptors.descList}  
 return descriptors  
 else:  
 return None  
  
# Apply the descriptor calculation function to the 'canonical\_smiles' column  
df['descriptors'] = df['canonical\_smiles'].apply(calculate\_descriptors)  
  
# Print the first few rows of the DataFrame  
print(df.head())

**Ví dụ 2: Phân tích SAR đơn giản (Simple SAR Analysis)**

* **SQL:** (Không cần thiết cho ví dụ này, dữ liệu đã có từ ví dụ 1)
* **Python:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Draw  
import os  
  
base\_path = "." # Thay đổi nếu cần  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Chuyển đổi cột standard\_value thành số  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce')  
  
# Lọc các phân tử có hoạt tính cao (IC50 < 100 nM)  
active\_molecules = df[df['standard\_value'] < 100]  
  
# In ra số lượng phân tử có hoạt tính cao  
print(f"Number of active molecules: {len(active\_molecules)}")  
  
# Vẽ một vài phân tử có hoạt tính cao  
molecules = [Chem.MolFromSmiles(smiles) for smiles in active\_molecules['canonical\_smiles'].head(4)]  
img = Draw.MolsToGridImage(molecules, molsPerRow=4, subImgSize=(200,200))  
img.save(os.path.join(base\_path, "data", "active\_molecules.png"))

* **English:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Draw  
import os  
  
base\_path = "." # Change if needed  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Convert the standard\_value column to numeric  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce')  
  
# Filter for highly active molecules (IC50 < 100 nM)  
active\_molecules = df[df['standard\_value'] < 100]  
  
# Print the number of active molecules  
print(f"Number of active molecules: {len(active\_molecules)}")  
  
# Draw a few highly active molecules  
molecules = [Chem.MolFromSmiles(smiles) for smiles in active\_molecules['canonical\_smiles'].head(4)]  
img = Draw.MolsToGridImage(molecules, molsPerRow=4, subImgSize=(200,200))  
img.save(os.path.join(base\_path, "data", "active\_molecules.png"))

**Ví dụ 3: Tính toán LogP và TPSA (Calculating LogP and TPSA)**

* **SQL:** (Không cần thiết cho ví dụ này, dữ liệu đã có từ ví dụ 1)
* **Python:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from rdkit.Chem import Crippen  
from rdkit.Chem import rdMolDescriptors  
import os  
  
base\_path = "." # Thay đổi nếu cần  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Hàm tính toán LogP và TPSA  
def calculate\_logp\_tpsa(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 logp = Crippen.MolLogP(mol)  
 tpsa = rdMolDescriptors.CalcTPSA(mol)  
 return logp, tpsa  
 else:  
 return None, None  
  
# Áp dụng hàm tính toán LogP và TPSA  
df[['logp', 'tpsa']] = df['canonical\_smiles'].apply(lambda x: pd.Series(calculate\_logp\_tpsa(x)))  
  
# In ra một vài dòng đầu của DataFrame  
print(df[['canonical\_smiles', 'logp', 'tpsa']].head())

* **English:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from rdkit.Chem import Crippen  
from rdkit.Chem import rdMolDescriptors  
import os  
  
base\_path = "." # Change if needed  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Function to calculate LogP and TPSA  
def calculate\_logp\_tpsa(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 logp = Crippen.MolLogP(mol)  
 tpsa = rdMolDescriptors.CalcTPSA(mol)  
 return logp, tpsa  
 else:  
 return None, None  
  
# Apply the LogP and TPSA calculation function  
df[['logp', 'tpsa']] = df['canonical\_smiles'].apply(lambda x: pd.Series(calculate\_logp\_tpsa(x)))  
  
# Print the first few rows of the DataFrame  
print(df[['canonical\_smiles', 'logp', 'tpsa']].head())

**Ví dụ 4: Xây dựng mô hình QSAR đơn giản (Building a Simple QSAR Model)**

* **SQL:** (Không cần thiết cho ví dụ này, dữ liệu đã có từ ví dụ 1 và 3)
* **Python:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
import numpy as np  
import os  
  
base\_path = "." # Thay đổi nếu cần  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Tính toán LogP và TPSA (nếu chưa có)  
def calculate\_logp\_tpsa(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 logp = Crippen.MolLogP(mol)  
 tpsa = rdMolDescriptors.CalcTPSA(mol)  
 return logp, tpsa  
 else:  
 return None, None  
  
df[['logp', 'tpsa']] = df['canonical\_smiles'].apply(lambda x: pd.Series(calculate\_logp\_tpsa(x)))  
  
# Chuyển đổi IC50 thành pIC50  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Chuyển đổi nM thành M  
  
# Chọn các thuộc tính (features) và biến mục tiêu (target)  
X = df[['logp', 'tpsa']].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()  
  
# Huấn luyện mô hình  
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)  
rmse = np.sqrt(mse)  
print(f"Root Mean Squared Error: {rmse}")

* **English:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
import numpy as np  
import os  
  
base\_path = "." # Change if needed  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Calculate LogP and TPSA (if not already calculated)  
def calculate\_logp\_tpsa(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 logp = Crippen.MolLogP(mol)  
 tpsa = rdMolDescriptors.CalcTPSA(mol)  
 return logp, tpsa  
 else:  
 return None, None  
  
df[['logp', 'tpsa']] = df['canonical\_smiles'].apply(lambda x: pd.Series(calculate\_logp\_tpsa(x)))  
  
# Convert IC50 to pIC50  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Convert nM to M  
  
# Select features and target variable  
X = df[['logp', 'tpsa']].fillna(0) # Fill NaN values with 0  
y = df['pIC50'].fillna(0)  
  
# Split data into training and test 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()  
  
# Train the model  
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)  
rmse = np.sqrt(mse)  
print(f"Root Mean Squared Error: {rmse}")

**Ví dụ 5: Phân tích cấu trúc con (Substructure Analysis)**

* **SQL:** (Không cần thiết cho ví dụ này, dữ liệu đã có từ ví dụ 1)
* **Python:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from rdkit.Chem import Draw  
from rdkit.Chem.Draw import IPythonConsole  
import os  
  
base\_path = "." # Thay đổi nếu cần  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Định nghĩa cấu trúc con (substructure) quan tâm (ví dụ: vòng benzen)  
substructure = Chem.MolFromSmiles('c1ccccc1')  
  
# Hàm kiểm tra xem một phân tử có chứa cấu trúc con hay không  
def has\_substructure(smiles, substructure):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return mol.HasSubstructMatch(substructure)  
 else:  
 return False  
  
# Tạo cột mới để chỉ ra sự hiện diện của cấu trúc con  
df['has\_benzene'] = df['canonical\_smiles'].apply(lambda x: has\_substructure(x, substructure))  
  
# Lọc các phân tử có chứa vòng benzen  
benzene\_molecules = df[df['has\_benzene'] == True]  
  
# In ra số lượng phân tử có chứa vòng benzen  
print(f"Number of molecules with benzene ring: {len(benzene\_molecules)}")  
  
# Vẽ một vài phân tử có chứa vòng benzen  
molecules = [Chem.MolFromSmiles(smiles) for smiles in benzene\_molecules['canonical\_smiles'].head(4)]  
img = Draw.MolsToGridImage(molecules, molsPerRow=4, subImgSize=(200,200))  
img.save(os.path.join(base\_path, "data", "benzene\_molecules.png"))

* **English:**

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from rdkit.Chem import Draw  
from rdkit.Chem.Draw import IPythonConsole  
import os  
  
base\_path = "." # Change if needed  
csv\_file\_path = os.path.join(base\_path, "data", "chembl\_activity\_data.csv")  
df = pd.read\_csv(csv\_file\_path)  
  
# Define the substructure of interest (e.g., benzene ring)  
substructure = Chem.MolFromSmiles('c1ccccc1')  
  
# Function to check if a molecule contains the substructure  
def has\_substructure(smiles, substructure):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return mol.HasSubstructMatch(substructure)  
 else:  
 return False  
  
# Create a new column to indicate the presence of the substructure  
df['has\_benzene'] = df['canonical\_smiles'].apply(lambda x: has\_substructure(x, substructure))  
  
# Filter for molecules containing the benzene ring  
benzene\_molecules = df[df['has\_benzene'] == True]  
  
# Print the number of molecules containing the benzene ring  
print(f"Number of molecules with benzene ring: {len(benzene\_molecules)}")  
  
# Draw a few molecules containing the benzene ring  
molecules = [Chem.MolFromSmiles(smiles) for smiles in benzene\_molecules['canonical\_smiles'].head(4)]  
img = Draw.MolsToGridImage(molecules, molsPerRow=4, subImgSize=(200,200))  
img.save(os.path.join(base\_path, "data", "benzene\_molecules.png"))

**Lưu ý quan trọng:**

* **Đường dẫn:** Hãy chắc chắn rằng bạn đã thay đổi base\_path để phù hợp với cấu trúc thư mục của bạn.
* **Cài đặt thư viện:** Đảm bảo bạn đã cài đặt tất cả các thư viện cần thiết (pandas, rdkit, scikit-learn).
* **Dữ liệu:** Các ví dụ này giả định rằng bạn đã tải dữ liệu từ ChEMBL và lưu nó vào file CSV. Bạn có thể cần điều chỉnh code để phù hợp với định dạng dữ liệu của bạn.
* **Phân tích sâu hơn:** Đây chỉ là những ví dụ cơ bản. Bạn có thể mở rộng chúng bằng cách sử dụng các thuộc tính hóa học khác, các mô hình máy học phức tạp hơn và các kỹ thuật phân tích dữ liệu khác.

Hãy cho tôi biết nếu bạn có bất kỳ câu hỏi nào hoặc cần hỗ trợ thêm. Chúc bạn thành công với dự án của mình!

1. 0-9. [↑](#footnote-ref-21)