# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Kiểm chứng chéo tương tác với DrugBank

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 sẽ cung cấp hướng dẫn chi tiết, code mẫu SQL và Python, cùng những lưu ý quan trọng để bạn hoàn thành dự án một cách hiệu quả.

**Tổng quan dự án:**

Dự án của bạn tập trung vào việc phân tích dữ liệu từ cơ sở dữ liệu ChEMBL 35, sử dụng RDKit để xử lý thông tin hóa học, và tuân theo cấu trúc thư mục chuẩn AIMLOps. Mục tiêu là trích xuất thông tin hữu ích cho quá trình nghiên cứu và phát triển thuốc.

**Phân tích mô hình (Analysis Model):**

Chúng ta sẽ tập trung vào việc xây dựng một quy trình làm việc (workflow) hoàn chỉnh từ việc truy vấn dữ liệu từ ChEMBL, tiền xử lý dữ liệu bằng RDKit, và cuối cùng là phân tích dữ liệu để trả lời các câu hỏi nghiên cứu cụ thể.

**Các bước chính:**

1. **Truy vấn dữ liệu từ ChEMBL (Data Retrieval):** Sử dụng SQL để trích xuất thông tin về các hợp chất và hoạt tính sinh học của chúng.
2. **Tiền xử lý dữ liệu (Data Preprocessing):** Sử dụng RDKit để chuyển đổi SMILES thành các đặc trưng hóa học (chemical features) có thể sử dụng trong các mô hình học máy.
3. **Phân tích dữ liệu (Data Analysis):** Sử dụng các kỹ thuật thống kê và học máy để khám phá các mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học.

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

* **English:** This project aims to analyze ChEMBL 35 data using RDKit within an AIMLOps framework. The goal is to extract valuable information for drug discovery and development. We will focus on building a complete workflow from data retrieval to analysis.
* **Tiếng Việt:** Dự án này nhằm mục đích phân tích dữ liệu ChEMBL 35 sử dụng RDKit trong khuôn khổ AIMLOps. Mục tiêu là trích xuất thông tin giá trị cho việc khám phá và phát triển thuốc. Chúng ta sẽ tập trung vào việc xây dựng một quy trình làm việc hoàn chỉnh từ việc truy xuất dữ liệu đến phân tích.

**Code SQL (SQL Code):**

Dưới đây là một số ví dụ SQL, lưu ý chỉ lấy 100 dòng để giảm tải cho máy tính.

*Ví dụ 1: Lấy thông tin cơ bản về các hợp chất có hoạt tính ức chế trên một mục tiêu cụ thể (e.g., EGFR).*

-- English  
-- Retrieve basic information about compounds with inhibitory activity on a specific target (e.g., EGFR)  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units,  
 act.standard\_type  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Epidermal Growth Factor Receptor' -- Replace with your target of interest  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Ensure standard\_value is numeric  
LIMIT 100;  
  
-- Tiếng Việt  
-- Lấy thông tin cơ bản về các hợp chất có hoạt tính ức chế trên một mục tiêu cụ thể (ví dụ: EGFR)  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units,  
 act.standard\_type  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Epidermal Growth Factor Receptor' -- Thay thế bằng mục tiêu bạn quan tâm  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Đảm bảo standard\_value là kiểu số  
LIMIT 100;

*Ví dụ 2: Lấy thông tin về các hợp chất có hoạt tính trên một protein cụ thể và lưu vào file CSV.*

-- English  
-- Retrieve information about compounds active on a specific protein and save to a CSV file.  
-- Requires you to run this in pgAdmin and export the result to a CSV file.  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units,  
 act.standard\_type  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Tyrosine-protein kinase ABL1' -- Example protein target  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Ensure standard\_value is numeric  
LIMIT 100;  
  
-- Tiếng Việt  
-- Lấy thông tin về các hợp chất có hoạt tính trên một protein cụ thể và lưu vào file CSV.  
-- Yêu cầu bạn chạy truy vấn này trong pgAdmin và xuất kết quả ra file CSV.  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units,  
 act.standard\_type  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Tyrosine-protein kinase ABL1' -- Ví dụ mục tiêu protein  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Đảm bảo standard\_value là kiểu số  
LIMIT 100;

*Ví dụ 3: Tìm kiếm các hợp chất có khối lượng phân tử nằm trong một khoảng nhất định.*

-- English  
-- Find compounds with molecular weight within a certain range.  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 md.mw\_freebase  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
WHERE  
 md.mw\_freebase BETWEEN 400 AND 600  
LIMIT 100;  
  
-- Tiếng Việt  
-- Tìm kiếm các hợp chất có khối lượng phân tử nằm trong một khoảng nhất định.  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 md.mw\_freebase  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
WHERE  
 md.mw\_freebase BETWEEN 400 AND 600  
LIMIT 100;

*Ví dụ 4: Lấy thông tin về các mục tiêu và số lượng hợp chất hoạt động trên mỗi mục tiêu.*

-- English  
-- Get information about targets and the number of compounds active on each target.  
SELECT  
 td.chembl\_id,  
 td.pref\_name,  
 COUNT(DISTINCT act.molregno) AS num\_compounds  
FROM  
 target\_dictionary td  
JOIN  
 activities act ON td.tid = act.tid  
WHERE act.standard\_value IS NOT NULL  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Ensure standard\_value is numeric  
GROUP BY  
 td.chembl\_id, td.pref\_name  
ORDER BY  
 num\_compounds DESC  
LIMIT 100;  
  
-- Tiếng Việt  
-- Lấy thông tin về các mục tiêu và số lượng hợp chất hoạt động trên mỗi mục tiêu.  
SELECT  
 td.chembl\_id,  
 td.pref\_name,  
 COUNT(DISTINCT act.molregno) AS num\_compounds  
FROM  
 target\_dictionary td  
JOIN  
 activities act ON td.tid = act.tid  
WHERE act.standard\_value IS NOT NULL  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Đảm bảo standard\_value là kiểu số  
GROUP BY  
 td.chembl\_id, td.pref\_name  
ORDER BY  
 num\_compounds DESC  
LIMIT 100;

*Ví dụ 5: Tìm kiếm các hợp chất có hoạt tính trên một mục tiêu cụ thể và có giá trị pChEMBL lớn hơn một giá trị nhất định.*

-- English  
-- Find compounds active on a specific target and have a pChEMBL value greater than a certain value.  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 act.pchembl\_value  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Cyclooxygenase-2' -- Example target  
 AND act.pchembl\_value > 6.0  
LIMIT 100;  
  
-- Tiếng Việt  
-- Tìm kiếm các hợp chất có hoạt tính trên một mục tiêu cụ thể và có giá trị pChEMBL lớn hơn một giá trị nhất định.  
SELECT DISTINCT  
 md.chembl\_id,  
 cs.canonical\_smiles,  
 act.pchembl\_value  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Cyclooxygenase-2' -- Ví dụ mục tiêu  
 AND act.pchembl\_value > 6.0  
LIMIT 100;

**Lưu ý về lỗi SQL (Note on SQL Error):**

Lỗi ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard\_value ~ '^[0-9\.]+$' xảy ra do kiểu dữ liệu của cột act.standard\_value không phải là numeric. Để khắc phục, bạn có thể thử ép kiểu dữ liệu trước khi so sánh:

AND act.standard\_value::numeric ~ '^[0-9\.]+$'

Hoặc sử dụng hàm REGEXP\_MATCHES (tùy thuộc vào phiên bản PostgreSQL của bạn).

**Code Python (Python Code):**

Dưới đây là ví dụ code Python sử dụng RDKit để đọc file CSV, tính toán fingerprint và chuẩn bị dữ liệu cho mô hình học máy.

# English  
import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import AllChem  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error  
  
# Define base path (replace with your actual path)  
base\_path = "../data"  
  
# Define file path for the CSV file  
csv\_file\_path = os.path.join(base\_path, "your\_data.csv") # Replace with your actual CSV file name  
  
# Load data from CSV  
try:  
 data = pd.read\_csv(csv\_file\_path)  
except FileNotFoundError:  
 print(f"Error: The file {csv\_file\_path} was not found.")  
 exit()  
  
# Data Cleaning and Preprocessing  
data = data.dropna(subset=['canonical\_smiles', 'standard\_value'])  
data = data[data['standard\_value'].astype(str).str.match(r'^[0-9\.]+$')] # Keep only numeric values  
data['standard\_value'] = pd.to\_numeric(data['standard\_value'], errors='coerce')  
data = data.dropna(subset=['standard\_value'])  
data = data.head(100)  
  
# RDKit Function  
def generate\_fingerprint(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)  
 return np.array(list(fp.ToBitString()), dtype=int)  
 else:  
 return None  
  
# Generate Fingerprints  
data['fingerprint'] = data['canonical\_smiles'].apply(generate\_fingerprint)  
data = data.dropna(subset=['fingerprint'])  
  
# Prepare Data for Machine Learning  
X = np.stack(data['fingerprint'].values)  
y = data['standard\_value'].values  
  
# Split Data  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Train Model  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
# Evaluate Model  
y\_pred = model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred)  
print(f"Mean Squared Error: {mse}")  
  
  
# Vietnamese  
import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import AllChem  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error  
  
# Định nghĩa đường dẫn gốc (thay thế bằng đường dẫn thực tế của bạn)  
base\_path = "../data"  
  
# Định nghĩa đường dẫn đến file CSV  
csv\_file\_path = os.path.join(base\_path, "your\_data.csv") # Thay thế bằng tên file CSV thực tế của bạn  
  
# Tải dữ liệu từ file CSV  
try:  
 data = pd.read\_csv(csv\_file\_path)  
except FileNotFoundError:  
 print(f"Lỗi: Không tìm thấy file {csv\_file\_path}.")  
 exit()  
  
# Làm sạch và tiền xử lý dữ liệu  
data = data.dropna(subset=['canonical\_smiles', 'standard\_value'])  
data = data[data['standard\_value'].astype(str).str.match(r'^[0-9\.]+$')] # Chỉ giữ lại các giá trị số  
data['standard\_value'] = pd.to\_numeric(data['standard\_value'], errors='coerce')  
data = data.dropna(subset=['standard\_value'])  
data = data.head(100)  
  
# Hàm RDKit  
def generate\_fingerprint(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)  
 return np.array(list(fp.ToBitString()), dtype=int)  
 else:  
 return None  
  
# Tạo Fingerprints  
data['fingerprint'] = data['canonical\_smiles'].apply(generate\_fingerprint)  
data = data.dropna(subset=['fingerprint'])  
  
# Chuẩn bị dữ liệu cho Học Máy  
X = np.stack(data['fingerprint'].values)  
y = data['standard\_value'].values  
  
# Chia Dữ Liệu  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Huấn Luyện Mô Hình  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
# Đánh Giá Mô Hình  
y\_pred = model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred)  
print(f"Lỗi Bình Phương Trung Bình: {mse}")

**Lưu ý về lỗi Python (Note on Python Error):**

Lỗi về tham số squared=False trong mean\_squared\_error chỉ xảy ra với các phiên bản scikit-learn rất cũ. Bạn nên nâng cấp scikit-learn lên phiên bản mới nhất để tránh lỗi này. Nếu không thể nâng cấp, bạn có thể tính căn bậc hai của MSE để có RMSE (Root Mean Squared Error).

**Các ví dụ Python khác:**

*Ví dụ 1: Tính các descriptor phân tử cơ bản (ví dụ: LogP, MW) sử dụng RDKit.*

# English  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 logp = Descriptors.MolLogP(mol)  
 mw = Descriptors.MolWt(mol)  
 return logp, mw  
 else:  
 return None, None  
  
data['logp'], data['mw'] = zip(\*data['canonical\_smiles'].apply(calculate\_descriptors))  
print(data[['canonical\_smiles', 'logp', 'mw']].head())  
  
# Vietnamese  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def tinh\_toan\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 logp = Descriptors.MolLogP(mol)  
 mw = Descriptors.MolWt(mol)  
 return logp, mw  
 else:  
 return None, None  
  
data['logp'], data['mw'] = zip(\*data['canonical\_smiles'].apply(tinh\_toan\_descriptors))  
print(data[['canonical\_smiles', 'logp', 'mw']].head())

*Ví dụ 2: Trực quan hóa một phân tử từ SMILES.*

# English  
from rdkit import Chem  
from rdkit.Chem.Draw import IPythonConsole  
from rdkit.Chem import Draw  
from IPython.display import SVG  
  
def visualize\_molecule(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return SVG(Draw.MolToSVG(mol, size=(300, 300)))  
 else:  
 return None  
  
smiles = data['canonical\_smiles'].iloc[0]  
visualize\_molecule(smiles)  
  
# Vietnamese  
from rdkit import Chem  
from rdkit.Chem.Draw import IPythonConsole  
from rdkit.Chem import Draw  
from IPython.display import SVG  
  
def truc\_quan\_hoa\_phan\_tu(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return SVG(Draw.MolToSVG(mol, size=(300, 300)))  
 else:  
 return None  
  
smiles = data['canonical\_smiles'].iloc[0]  
truc\_quan\_hoa\_phan\_tu(smiles)

*Ví dụ 3: Tạo và trực quan hóa fingerprint (ví dụ: Morgan fingerprint).*

# English  
from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.Chem import Draw  
from IPython.display import SVG  
  
def visualize\_fingerprint(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 fp = AllChem.GetMorganFingerprint(mol, 2)  
 return fp  
 else:  
 return None  
  
fp = visualize\_fingerprint(data['canonical\_smiles'].iloc[0])  
print(fp)  
  
# Vietnamese  
from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.Chem import Draw  
from IPython.display import SVG  
  
def truc\_quan\_hoa\_fingerprint(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 fp = AllChem.GetMorganFingerprint(mol, 2)  
 return fp  
 else:  
 return None  
  
fp = truc\_quan\_hoa\_fingerprint(data['canonical\_smiles'].iloc[0])  
print(fp)

*Ví dụ 4: Sử dụng PCA để giảm chiều dữ liệu fingerprint.*

# English  
from sklearn.decomposition import PCA  
import matplotlib.pyplot as plt  
  
# Assuming X is the fingerprint matrix from the previous example  
pca = PCA(n\_components=2)  
X\_pca = pca.fit\_transform(X)  
  
plt.scatter(X\_pca[:, 0], X\_pca[:, 1])  
plt.xlabel('PCA Component 1')  
plt.ylabel('PCA Component 2')  
plt.title('PCA of Fingerprints')  
plt.show()  
  
# Vietnamese  
from sklearn.decomposition import PCA  
import matplotlib.pyplot as plt  
  
# Giả sử X là ma trận fingerprint từ ví dụ trước  
pca = PCA(n\_components=2)  
X\_pca = pca.fit\_transform(X)  
  
plt.scatter(X\_pca[:, 0], X\_pca[:, 1])  
plt.xlabel('Thành phần PCA 1')  
plt.ylabel('Thành phần PCA 2')  
plt.title('PCA của Fingerprints')  
plt.show()

*Ví dụ 5: Xây dựng mô hình Random Forest để dự đoán hoạt tính.*

# English  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
y\_pred = model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred)  
print(f"Mean Squared Error: {mse}")  
  
# Vietnamese  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
y\_pred = model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred)  
print(f"Lỗi Bình Phương Trung Bình: {mse}")

**Cấu trúc thư mục (Folder Structure):**

Đảm bảo rằng bạn tuân thủ cấu trúc thư mục AIMLOps:

├── data/  
│ └── your\_data.csv # Dữ liệu CSV từ ChEMBL  
├── notebooks/  
│ ├── Topic\_CheMBL\_35\_83\_1\_data\_extraction.ipynb  
│ ├── Topic\_CheMBL\_35\_83\_2\_data\_analysis.ipynb  
├── models/  
│ └── # Lưu trữ các mô hình đã huấn luyện  
├── scripts/  
│ └── # Lưu trữ các script Python  
└── README.md

**Lời khuyên (Advice):**

* **Chia nhỏ các bước:** Chia dự án thành các phần nhỏ hơn, dễ quản lý hơn (ví dụ: trích xuất dữ liệu, tiền xử lý, phân tích).
* **Ghi lại quá trình:** Ghi lại tất cả các bước thực hiện, các quyết định và kết quả. Điều này rất quan trọng để tái tạo và gỡ lỗi.
* **Kiểm tra dữ liệu:** Luôn kiểm tra dữ liệu của bạn để đảm bảo tính chính xác và đầy đủ.
* **Sử dụng virtual environment:** Sử dụng virtual environment để quản lý các thư viện Python và tránh xung đột.

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