# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Vẽ 3D không gian hóa học phân tử

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 nghiên cứu và phát triển dược học, tôi hiểu rõ tầm quan trọng của việc khai thác dữ liệu ChEMBL để tìm kiếm các hợp chất tiềm năng. Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu SQL và Python, cùng với các ví dụ để bạn bắt đầu.

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

Dự án của bạn tập trung vào việc phân tích dữ liệu ChEMBL 35, một cơ sở dữ liệu lớn chứa thông tin về các phân tử hoạt tính sinh học. Mục tiêu là sử dụng RDKit để trích xuất các đặc trưng hóa học từ cấu trúc phân tử và kết hợp chúng với dữ liệu hoạt tính sinh học để xây dựng mô hình dự đoán.

* **Data Source (Nguồn dữ liệu):** ChEMBL 35 database.
* **Tools (Công cụ):**
  + psql (PostgreSQL) for database querying.
  + RDKit for chemical feature extraction.
  + Jupyter Notebook for code execution and documentation.
  + Scikit-learn (or other machine learning libraries) for model building.
* **Steps (Các bước):**
  1. **Data Extraction (Trích xuất dữ liệu):** Use SQL queries to extract relevant data from ChEMBL, such as compound structures (SMILES) and activity data (IC50, Ki, etc.).
  2. **Feature Generation (Tạo đặc trưng):** Use RDKit to generate molecular descriptors (e.g., fingerprints, physicochemical properties) from the SMILES strings.
  3. **Data Preprocessing (Tiền xử lý dữ liệu):** Clean and prepare the data for modeling. This may involve handling missing values, scaling features, and splitting the data into training and testing sets.
  4. **Model Building (Xây dựng mô hình):** Train a machine learning model to predict activity based on the molecular descriptors. Common models include Random Forest, Support Vector Machines, and Neural Networks.
  5. **Model Evaluation (Đánh giá mô hình):** Evaluate the performance of the model using appropriate metrics (e.g., AUC, RMSE, R-squared).
  6. **Analysis and Interpretation (Phân tích và giải thích):** Analyze the model to identify important features and gain insights into the structure-activity relationship.

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

Dưới đây là hướng dẫn chi tiết bằng cả tiếng Anh và tiếng Việt:

* **English:**
  1. **Connect to the ChEMBL database using psql.**
  2. **Execute SQL queries to extract the desired data (e.g., compound structures and activity data).**
  3. **Save the extracted data as CSV files in the ../data/ directory.**
  4. **Load the CSV files into a Jupyter Notebook.**
  5. **Use RDKit to generate molecular descriptors from the SMILES strings.**
  6. **Preprocess the data (handle missing values, scale features, etc.).**
  7. **Split the data into training and testing sets.**
  8. **Train a machine learning model to predict activity.**
  9. **Evaluate the model’s performance.**
  10. **Analyze the model to identify important features and gain insights into the structure-activity relationship.**
* **Tiếng Việt:**
  1. **Kết nối đến cơ sở dữ liệu ChEMBL bằng psql.**
  2. **Thực hiện các truy vấn SQL để trích xuất dữ liệu mong muốn (ví dụ: cấu trúc hợp chất và dữ liệu hoạt tính).**
  3. **Lưu dữ liệu đã trích xuất dưới dạng tệp CSV trong thư mục ../data/.**
  4. **Tải các tệp CSV vào Jupyter Notebook.**
  5. **Sử dụng RDKit để tạo các đặc trưng phân tử từ chuỗi SMILES.**
  6. **Tiền xử lý dữ liệu (xử lý giá trị thiếu, chia tỷ lệ đặc trưng, v.v.).**
  7. **Chia dữ liệu thành các tập huấn luyện và kiểm tra.**
  8. **Huấn luyện mô hình học máy để dự đoán hoạt tính.**
  9. **Đánh giá hiệu suất của mô hình.**
  10. **Phân tích mô hình để xác định các đặc trưng quan trọng và hiểu rõ hơn về mối quan hệ cấu trúc-hoạt tính.**

**3. Code SQL và Python mẫu (Sample SQL and Python Code)**

**SQL (English):**

-- Extract 100 compounds with IC50 values for a specific target  
SELECT  
 molregno,  
 compound\_structures.canonical\_smiles,  
 activities.standard\_value,  
 activities.standard\_units  
FROM  
 compound\_structures  
JOIN  
 activities ON compound\_structures.molregno = activities.molregno  
JOIN  
 target\_dictionary ON activities.tid = target\_dictionary.tid  
WHERE  
 target\_dictionary.pref\_name = 'desired\_target\_name' -- Replace with your target  
 AND activities.standard\_type = 'IC50'  
 AND activities.standard\_units = 'nM'  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Only numeric values  
LIMIT 100;

**SQL (Tiếng Việt):**

-- Trích xuất 100 hợp chất với giá trị IC50 cho một mục tiêu cụ thể  
SELECT  
 molregno,  
 compound\_structures.canonical\_smiles,  
 activities.standard\_value,  
 activities.standard\_units  
FROM  
 compound\_structures  
JOIN  
 activities ON compound\_structures.molregno = activities.molregno  
JOIN  
 target\_dictionary ON activities.tid = target\_dictionary.tid  
WHERE  
 target\_dictionary.pref\_name = 'desired\_target\_name' -- Thay thế bằng mục tiêu của bạn  
 AND activities.standard\_type = 'IC50'  
 AND activities.standard\_units = 'nM'  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Chỉ giá trị số  
LIMIT 100;

**Python (English):**

import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.Chem import Descriptors  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Define the base path  
base\_path = "../data/"  
  
# Load the data from CSV  
data = pd.read\_csv(os.path.join(base\_path, "your\_data.csv")) # Replace with your CSV file  
  
# Function to calculate molecular descriptors  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 descriptors = {}  
 descriptors['MolLogP'] = Descriptors.MolLogP(mol)  
 descriptors['MolWt'] = Descriptors.MolWt(mol)  
 descriptors['NumHAcceptors'] = Descriptors.NumHAcceptors(mol)  
 descriptors['NumHDonors'] = Descriptors.NumHDonors(mol)  
 return descriptors  
 else:  
 return None  
  
# Apply the function to each SMILES string  
data['descriptors'] = data['canonical\_smiles'].apply(calculate\_descriptors)  
  
# Handle missing descriptors  
data = data.dropna(subset=['descriptors'])  
  
# Convert descriptors to DataFrame  
descriptors\_df = pd.DataFrame(data['descriptors'].tolist())  
data = pd.concat([data, descriptors\_df], axis=1)  
  
# Prepare data for modeling  
X = data[['MolLogP', 'MolWt', 'NumHAcceptors', 'NumHDonors']] # Use only available descriptors  
y = data['standard\_value']  
  
# 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)  
  
# Train a Random Forest Regressor model  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
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)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")

**Python (Tiếng Việt):**

import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.Chem import Descriptors  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Xác định đường dẫn cơ sở  
base\_path = "../data/"  
  
# Tải dữ liệu từ tệp CSV  
data = pd.read\_csv(os.path.join(base\_path, "your\_data.csv")) # Thay thế bằng tệp CSV của bạn  
  
# Hàm tính toán các descriptor phân tử  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 descriptors = {}  
 descriptors['MolLogP'] = Descriptors.MolLogP(mol)  
 descriptors['MolWt'] = Descriptors.MolWt(mol)  
 descriptors['NumHAcceptors'] = Descriptors.NumHAcceptors(mol)  
 descriptors['NumHDonors'] = Descriptors.NumHDonors(mol)  
 return descriptors  
 else:  
 return None  
  
# Áp dụng hàm cho mỗi chuỗi SMILES  
data['descriptors'] = data['canonical\_smiles'].apply(calculate\_descriptors)  
  
# Xử lý các descriptor bị thiếu  
data = data.dropna(subset=['descriptors'])  
  
# Chuyển đổi descriptors thành DataFrame  
descriptors\_df = pd.DataFrame(data['descriptors'].tolist())  
data = pd.concat([data, descriptors\_df], axis=1)  
  
# Chuẩn bị dữ liệu cho mô hình hóa  
X = data[['MolLogP', 'MolWt', 'NumHAcceptors', 'NumHDonors']] # Chỉ sử dụng các descriptor có sẵn  
y = data['standard\_value']  
  
# Chia dữ liệu thành các tập huấn luyện và kiểm tra  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Huấn luyện mô hình Random Forest Regressor  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
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)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")

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

Dưới đây là 5 ví dụ về code SQL và Python để bạn tham khảo:

**Example 1: Extract compounds with specific activity range**

* **SQL (English):**

SELECT molregno, compound\_structures.canonical\_smiles, activities.standard\_value  
FROM compound\_structures  
JOIN activities ON compound\_structures.molregno = activities.molregno  
WHERE activities.standard\_type = 'IC50'  
AND activities.standard\_units = 'nM'  
AND activities.standard\_value >= 100 AND activities.standard\_value <= 1000  
LIMIT 100;

* **SQL (Tiếng Việt):**

SELECT molregno, compound\_structures.canonical\_smiles, activities.standard\_value  
FROM compound\_structures  
JOIN activities ON compound\_structures.molregno = activities.molregno  
WHERE activities.standard\_type = 'IC50'  
AND activities.standard\_units = 'nM'  
AND activities.standard\_value >= 100 AND activities.standard\_value <= 1000  
LIMIT 100;

* **Python (English):**

# Read the CSV file  
df = pd.read\_csv(os.path.join(base\_path, "activity\_range.csv"))  
  
# Print the number of rows and columns  
print(f"Number of rows: {len(df)}")  
print(f"Number of columns: {len(df.columns)}")  
  
# Print the first 5 rows of the DataFrame  
print(df.head())

* **Python (Tiếng Việt):**

# Đọc file CSV  
df = pd.read\_csv(os.path.join(base\_path, "activity\_range.csv"))  
  
# In số lượng hàng và cột  
print(f"Số lượng hàng: {len(df)}")  
print(f"Số lượng cột: {len(df.columns)}")  
  
# In 5 hàng đầu tiên của DataFrame  
print(df.head())

**Example 2: Calculate Lipinski’s Rule of Five**

* **Python (English):**

def lipinski\_rule\_of\_five(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 mw = Descriptors.MolWt(mol)  
 logp = Descriptors.MolLogP(mol)  
 hbd = Descriptors.NumHDonors(mol)  
 hba = Descriptors.NumHAcceptors(mol)  
  
 # Check Lipinski's rule of five  
 rule\_1 = mw <= 500  
 rule\_2 = logp <= 5  
 rule\_3 = hbd <= 5  
 rule\_4 = hba <= 10  
  
 # Count number of rules that fail  
 num\_failed\_rules = sum([not rule\_1, not rule\_2, not rule\_3, not rule\_4])  
 return num\_failed\_rules  
 else:  
 return None  
  
# Apply Lipinski's Rule of Five to each SMILES  
df['Lipinski\_Failures'] = df['canonical\_smiles'].apply(lipinski\_rule\_of\_five)  
  
# Print the results  
print(df[['canonical\_smiles', 'Lipinski\_Failures']].head())

* **Python (Tiếng Việt):**

def lipinski\_rule\_of\_five(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 mw = Descriptors.MolWt(mol)  
 logp = Descriptors.MolLogP(mol)  
 hbd = Descriptors.NumHDonors(mol)  
 hba = Descriptors.NumHAcceptors(mol)  
  
 # Kiểm tra quy tắc 5 của Lipinski  
 rule\_1 = mw <= 500  
 rule\_2 = logp <= 5  
 rule\_3 = hbd <= 5  
 rule\_4 = hba <= 10  
  
 # Đếm số lượng quy tắc không đạt  
 num\_failed\_rules = sum([not rule\_1, not rule\_2, not rule\_3, not rule\_4])  
 return num\_failed\_rules  
 else:  
 return None  
  
# Áp dụng quy tắc 5 của Lipinski cho mỗi SMILES  
df['Lipinski\_Failures'] = df['canonical\_smiles'].apply(lipinski\_rule\_of\_five)  
  
# In kết quả  
print(df[['canonical\_smiles', 'Lipinski\_Failures']].head())

**Example 3: Extract compounds based on substructure**

* **SQL (English):**

SELECT molregno, compound\_structures.canonical\_smiles  
FROM compound\_structures  
WHERE compound\_structures.canonical\_smiles LIKE '%C=O%' -- Example: compounds containing a carbonyl group  
LIMIT 100;

* **SQL (Tiếng Việt):**

SELECT molregno, compound\_structures.canonical\_smiles  
FROM compound\_structures  
WHERE compound\_structures.canonical\_smiles LIKE '%C=O%' -- Ví dụ: hợp chất chứa nhóm carbonyl  
LIMIT 100;

* **Python (English):**

from rdkit import Chem  
  
def check\_substructure(smiles, substructure\_smarts):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 substructure = Chem.MolFromSmarts(substructure\_smarts)  
 if substructure is not None:  
 return mol.HasSubstructMatch(substructure)  
 else:  
 return False  
 else:  
 return False  
  
# Define substructure SMARTS  
substructure\_smarts = "C=O" # Carbonyl group  
  
# Apply the function to each SMILES  
df['Has\_Carbonyl'] = df['canonical\_smiles'].apply(lambda x: check\_substructure(x, substructure\_smarts))  
  
# Print the results  
print(df[['canonical\_smiles', 'Has\_Carbonyl']].head())

* **Python (Tiếng Việt):**

from rdkit import Chem  
  
def check\_substructure(smiles, substructure\_smarts):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 substructure = Chem.MolFromSmarts(substructure\_smarts)  
 if substructure is not None:  
 return mol.HasSubstructMatch(substructure)  
 else:  
 return False  
 else:  
 return False  
  
# Xác định cấu trúc con SMARTS  
substructure\_smarts = "C=O" # Nhóm carbonyl  
  
# Áp dụng hàm cho mỗi SMILES  
df['Has\_Carbonyl'] = df['canonical\_smiles'].apply(lambda x: check\_substructure(x, substructure\_smarts))  
  
# In kết quả  
print(df[['canonical\_smiles', 'Has\_Carbonyl']].head())

**Example 4: Calculate Molecular Weight**

* **Python (English):**

from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def calculate\_molecular\_weight(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return Descriptors.MolWt(mol)  
 else:  
 return None  
  
# Apply the function to each SMILES  
df['Molecular\_Weight'] = df['canonical\_smiles'].apply(calculate\_molecular\_weight)  
  
# Print the results  
print(df[['canonical\_smiles', 'Molecular\_Weight']].head())

* **Python (Tiếng Việt):**

from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def calculate\_molecular\_weight(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return Descriptors.MolWt(mol)  
 else:  
 return None  
  
# Áp dụng hàm cho mỗi SMILES  
df['Molecular\_Weight'] = df['canonical\_smiles'].apply(calculate\_molecular\_weight)  
  
# In kết quả  
print(df[['canonical\_smiles', 'Molecular\_Weight']].head())

**Example 5: Visualize Molecules**

* **Python (English):**

from rdkit import Chem  
from rdkit.Chem.Draw import IPythonConsole  
from rdkit.Chem import Draw  
  
def visualize\_molecule(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return Draw.MolToImage(mol)  
 else:  
 return None  
  
# Visualize the first molecule  
img = visualize\_molecule(df['canonical\_smiles'].iloc[0])  
img

* **Python (Tiếng Việt):**

from rdkit import Chem  
from rdkit.Chem.Draw import IPythonConsole  
from rdkit.Chem import Draw  
  
def visualize\_molecule(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return Draw.MolToImage(mol)  
 else:  
 return None  
  
# Visualize phân tử đầu tiên  
img = visualize\_molecule(df['canonical\_smiles'].iloc[0])  
img

**5. Giải quyết lỗi (Error Resolution)**

* **ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard\_value ~ ‘**[[1]](#footnote-21)**+$’**
  + **Explanation (Giải thích):** Lỗi này xảy ra vì toán tử ~ (regex match) không được hỗ trợ trực tiếp trên kiểu dữ liệu numeric trong PostgreSQL.
  + **Solution (Giải pháp):** Chuyển đổi cột standard\_value sang kiểu text trước khi so sánh với biểu thức chính quy.
  + **SQL (English):**

SELECT  
 molregno,  
 compound\_structures.canonical\_smiles,  
 activities.standard\_value,  
 activities.standard\_units  
FROM  
 compound\_structures  
JOIN  
 activities ON compound\_structures.molregno = activities.molregno  
JOIN  
 target\_dictionary ON activities.tid = target\_dictionary.tid  
WHERE  
 target\_dictionary.pref\_name = 'desired\_target\_name' -- Replace with your target  
 AND activities.standard\_type = 'IC50'  
 AND activities.standard\_units = 'nM'  
 AND CAST(activities.standard\_value AS TEXT) ~ '^[0-9\.]+$' -- Convert to TEXT  
LIMIT 100;

\* \*\*SQL (Tiếng Việt):\*\*

SELECT  
 molregno,  
 compound\_structures.canonical\_smiles,  
 activities.standard\_value,  
 activities.standard\_units  
FROM  
 compound\_structures  
JOIN  
 activities ON compound\_structures.molregno = activities.molregno  
JOIN  
 target\_dictionary ON activities.tid = target\_dictionary.tid  
WHERE  
 target\_dictionary.pref\_name = 'desired\_target\_name' -- Thay thế bằng mục tiêu của bạn  
 AND activities.standard\_type = 'IC50'  
 AND activities.standard\_units = 'nM'  
 AND CAST(activities.standard\_value AS TEXT) ~ '^[0-9\.]+$' -- Chuyển đổi sang TEXT  
LIMIT 100;

* **Phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean\_squared\_error**
  + **Explanation (Giải thích):** Tham số squared=False được thêm vào hàm mean\_squared\_error trong các phiên bản scikit-learn mới hơn để trả về Root Mean Squared Error (RMSE) thay vì Mean Squared Error (MSE).
  + **Solution (Giải pháp):**
    1. **Update scikit-learn (Cập nhật scikit-learn):** Nâng cấp phiên bản scikit-learn của bạn lên phiên bản mới nhất bằng lệnh pip install --upgrade scikit-learn.
    2. **Calculate RMSE manually (Tính RMSE thủ công):** Nếu bạn không thể nâng cấp scikit-learn, bạn có thể tính RMSE bằng cách lấy căn bậc hai của MSE.
    - **Python (English):**

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

\* \*\*Python (Tiếng Việt):\*\*

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

**Lưu ý quan trọng:**

* Hãy thay thế "your\_data.csv" và "desired\_target\_name" bằng tên tệp và mục tiêu cụ thể của bạn.
* Đảm bảo rằng bạn đã cài đặt tất cả các thư viện cần thiết (RDKit, pandas, scikit-learn, etc.) trước khi chạy code Python.
* Khi làm việc với dữ liệu lớn, hãy cân nhắc sử dụng các kỹ thuật tối ưu hóa để cải thiện hiệu suất.

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.

1. 0-9. [↑](#footnote-ref-21)