# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Phân tử kháng virus tương ứng với đích COVID-19

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 chuyên gia nghiên cứu và phát triển dược học, tôi sẽ cung cấp cho bạn những phân tích, hướng dẫn và mã nguồn mẫu để bạn có thể tiến hành dự án một cách hiệu quả.

**Phân tích tổng quan**

Dự án của bạn tập trung vào việc khai thác dữ liệu từ cơ sở dữ liệu ChEMBL 35, một nguồn tài nguyên vô giá trong lĩnh vực phát triển thuốc. Việc kết hợp ChEMBL với RDKit, một bộ công cụ hóa tin học mạnh mẽ, cho phép bạn thực hiện các phân tích sâu sắc về cấu trúc phân tử, tính chất hóa lý và hoạt tính sinh học của các hợp chất.

**Mục tiêu của dự án:**

* **Thu thập dữ liệu:** Lấy dữ liệu liên quan từ cơ sở dữ liệu ChEMBL 35 bằng các truy vấn SQL.
* **Tiền xử lý dữ liệu:** Làm sạch và chuẩn hóa dữ liệu, loại bỏ các giá trị ngoại lệ và xử lý các giá trị thiếu.
* **Tính toán các descriptor phân tử:** Sử dụng RDKit để tính toán các descriptor phân tử, mô tả các đặc tính cấu trúc và hóa lý của các hợp chất.
* **Phân tích dữ liệu:** Áp dụng các kỹ thuật thống kê và học máy để khám phá các mối quan hệ giữa cấu trúc phân tử, tính chất và hoạt tính sinh học.
* **Xây dựng mô hình:** Xây dựng các mô hình dự đoán hoạt tính sinh học của các hợp chất dựa trên cấu trúc của chúng.
* **Trực quan hóa dữ liệu:** Sử dụng các công cụ trực quan hóa để trình bày kết quả phân tích một cách rõ ràng và dễ hiểu.

**1. Phân tích và mô hình phân tích**

Chúng ta sẽ tập trung vào việc xây dựng một quy trình phân tích dữ liệu ChEMBL 35 để dự đoán hoạt tính sinh học của các hợp chất. Dưới đây là mô hình phân tích chi tiết:

**Bước 1: Thu thập dữ liệu (Data Acquisition)**

* **Mục tiêu:** Lấy dữ liệu từ cơ sở dữ liệu ChEMBL 35, bao gồm thông tin về cấu trúc phân tử (SMILES), hoạt tính sinh học (IC50, Ki, EC50, v.v.) và các thuộc tính liên quan khác.
* **Công cụ:** SQL, psycopg2 (Python library for PostgreSQL).
* **Đầu ra:** Một tập dữ liệu chứa thông tin về các hợp chất và hoạt tính sinh học của chúng.

**Bước 2: Tiền xử lý dữ liệu (Data Preprocessing)**

* **Mục tiêu:** Làm sạch và chuẩn hóa dữ liệu để đảm bảo chất lượng và tính nhất quán.
* **Công cụ:** Pandas (Python library for data manipulation), RDKit.
* **Các bước:**
  + Loại bỏ các hợp chất trùng lặp.
  + Chuẩn hóa chuỗi SMILES.
  + Xử lý các giá trị hoạt tính sinh học (ví dụ: chuyển đổi IC50 sang pIC50).
  + Loại bỏ các giá trị ngoại lệ hoặc không hợp lệ.

**Bước 3: Tính toán descriptor phân tử (Molecular Descriptor Calculation)**

* **Mục tiêu:** Tính toán các descriptor phân tử từ cấu trúc của các hợp chất.
* **Công cụ:** RDKit.
* **Các descriptor:**
  + Các descriptor 2D: Molecular Weight, LogP, Topological Polar Surface Area (TPSA).
  + Các descriptor dựa trên fragments: Murcko Fragments.
  + Các descriptor khác: Số lượng vòng, số lượng nguyên tử, v.v.

**Bước 4: Phân tích dữ liệu và xây dựng mô hình (Data Analysis and Model Building)**

* **Mục tiêu:** Xây dựng các mô hình học máy để dự đoán hoạt tính sinh học của các hợp chất dựa trên các descriptor phân tử.
* **Công cụ:** Scikit-learn (Python library for machine learning).
* **Các thuật toán:**
  + Hồi quy tuyến tính (Linear Regression).
  + Máy vector hỗ trợ (Support Vector Machine - SVM).
  + Rừng ngẫu nhiên (Random Forest).
  + Mạng nơ-ron (Neural Networks).
* **Đánh giá mô hình:** Sử dụng các kỹ thuật đánh giá mô hình phù hợp (ví dụ: cross-validation, R-squared, RMSE) để đánh giá hiệu suất của các mô hình.

**Bước 5: Trực quan hóa dữ liệu (Data Visualization)**

* **Mục tiêu:** Trình bày kết quả phân tích một cách rõ ràng và dễ hiểu.
* **Công cụ:** Matplotlib, Seaborn (Python libraries for data visualization).
* **Các loại biểu đồ:**
  + Biểu đồ phân tán (Scatter plots) để hiển thị mối quan hệ giữa các descriptor và hoạt tính sinh học.
  + Biểu đồ hộp (Box plots) để so sánh phân phối của các descriptor giữa các nhóm hợp chất khác nhau.
  + Biểu đồ quan trọng tính năng (Feature importance plots) để xác định các descriptor quan trọng nhất trong mô hình.

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

Dưới đây là hướng dẫn chi tiết bằng cả tiếng Anh và tiếng Việt cho từng bước trong quy trình phân tích:

**Bước 1: Thu thập dữ liệu (Data Acquisition)**

* **English:**
  1. Connect to the ChEMBL 35 database using psycopg2.
  2. Execute SQL queries to retrieve the necessary data (e.g., compound structures, activity data).
  3. Save the data to a CSV file.
* **Tiếng Việt:**
  1. Kết nối đến cơ sở dữ liệu ChEMBL 35 bằng psycopg2.
  2. Thực thi các truy vấn SQL để lấy dữ liệu cần thiết (ví dụ: cấu trúc hợp chất, dữ liệu hoạt tính).
  3. Lưu dữ liệu vào một file CSV.

**Bước 2: Tiền xử lý dữ liệu (Data Preprocessing)**

* **English:**
  1. Load the CSV file into a Pandas DataFrame.
  2. Remove duplicate compounds.
  3. Standardize SMILES strings using RDKit.
  4. Convert activity values (e.g., IC50) to pIC50.
  5. Remove outliers or invalid values.
* **Tiếng Việt:**
  1. Tải file CSV vào một Pandas DataFrame.
  2. Loại bỏ các hợp chất trùng lặp.
  3. Chuẩn hóa chuỗi SMILES bằng RDKit.
  4. Chuyển đổi giá trị hoạt tính (ví dụ: IC50) sang pIC50.
  5. Loại bỏ các giá trị ngoại lệ hoặc không hợp lệ.

**Bước 3: Tính toán descriptor phân tử (Molecular Descriptor Calculation)**

* **English:**
  1. Use RDKit to calculate molecular descriptors for each compound.
  2. Calculate 2D descriptors (e.g., Molecular Weight, LogP, TPSA).
  3. Calculate fragment-based descriptors (e.g., Murcko Fragments).
  4. Add the descriptors to the Pandas DataFrame.
* **Tiếng Việt:**
  1. Sử dụng RDKit để tính toán các descriptor phân tử cho mỗi hợp chất.
  2. Tính toán các descriptor 2D (ví dụ: Molecular Weight, LogP, TPSA).
  3. Tính toán các descriptor dựa trên fragments (ví dụ: Murcko Fragments).
  4. Thêm các descriptor vào Pandas DataFrame.

**Bước 4: Phân tích dữ liệu và xây dựng mô hình (Data Analysis and Model Building)**

* **English:**
  1. Split the data into training and testing sets.
  2. Train a machine learning model (e.g., Linear Regression, SVM, Random Forest) using the training data.
  3. Evaluate the model’s performance on the testing data.
* **Tiếng Việt:**
  1. Chia dữ liệu thành tập huấn luyện và tập kiểm tra.
  2. Huấn luyện một mô hình học máy (ví dụ: Linear Regression, SVM, Random Forest) sử dụng dữ liệu huấn luyện.
  3. Đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra.

**Bước 5: Trực quan hóa dữ liệu (Data Visualization)**

* **English:**
  1. Use Matplotlib and Seaborn to create visualizations of the data and model results.
  2. Create scatter plots, box plots, and feature importance plots.
* **Tiếng Việt:**
  1. Sử dụng Matplotlib và Seaborn để tạo các hình ảnh trực quan về dữ liệu và kết quả mô hình.
  2. Tạo biểu đồ phân tán, biểu đồ hộp và biểu đồ quan trọng tính năng.

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

Dưới đây là các ví dụ về code SQL và Python để thực hiện các bước trong quy trình phân tích. Tôi sẽ tập trung vào việc khắc phục các lỗi bạn đã đề cập.

**a. SQL Code (Data Acquisition)**

-- English  
-- Select 100 compounds with activity data  
SELECT DISTINCT ON (molregno)  
 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\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9\.]+$' -- Corrected line  
ORDER BY molregno  
LIMIT 100;  
  
-- Vietnamese  
-- Chọn 100 hợp chất có dữ liệu hoạt tính  
SELECT DISTINCT ON (molregno)  
 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' -- Bạn có thể thay đổi thành loại hoạt tính khác  
 AND act.standard\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9\.]+$' -- Dòng đã sửa  
ORDER BY molregno  
LIMIT 100;

**Giải thích lỗi và cách sửa:**

* **Lỗi:** ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard\_value ~ '^[0-9\.]+$'
* **Nguyên nhân:** Toán tử ~ trong PostgreSQL được sử dụng để so sánh một chuỗi với một regular expression. Trong trường hợp này, act.standard\_value có kiểu dữ liệu là numeric, không phải là text.
* **Cách sửa:** Ép kiểu act.standard\_value về text bằng cách sử dụng act.standard\_value::text.

**b. Python Code (Data Preprocessing and Descriptor Calculation)**

# English  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import numpy as np  
import os  
  
base\_path = "../data" # Adjust as needed  
  
def calculate\_descriptors(smiles):  
 """Calculates molecular descriptors using RDKit."""  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 descriptors = {}  
 descriptors['Molecular Weight'] = Descriptors.MolWt(mol)  
 descriptors['LogP'] = Chem.Crippen.MolLogP(mol)  
 descriptors['TPSA'] = Chem.rdMolDescriptors.CalcTPSA(mol)  
 return descriptors  
  
# Load data  
data = pd.read\_csv(os.path.join(base\_path, "chembl\_data.csv")) # Replace with your CSV file  
  
# Handle missing SMILES  
data = data.dropna(subset=['canonical\_smiles'])  
  
# Calculate descriptors  
data['descriptors'] = data['canonical\_smiles'].apply(calculate\_descriptors)  
data = data.dropna(subset=['descriptors'])  
data = data[data['descriptors'].notna()]  
  
# Convert descriptors to columns  
data = pd.concat([data.drop(['descriptors'], axis=1), data['descriptors'].apply(pd.Series)], axis=1)  
  
# Convert IC50 to pIC50  
def ic50\_to\_pic50(ic50):  
 """Converts IC50 to pIC50."""  
 if pd.isna(ic50):  
 return np.nan  
 ic50 = float(ic50)  
 if ic50 <= 0:  
 return np.nan  
 pIC50 = -np.log10(ic50 / 1e9)  
 return pIC50  
  
data['pIC50'] = data['standard\_value'].apply(ic50\_to\_pic50)  
  
# Display the first few rows  
print(data.head())  
  
  
# Vietnamese  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import numpy as np  
import os  
  
base\_path = "../data" # Điều chỉnh nếu cần  
  
def tinh\_toan\_descriptor(smiles):  
 """Tính toán các descriptor phân tử sử dụng RDKit."""  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
 descriptors = {}  
 descriptors['Molecular Weight'] = Descriptors.MolWt(mol)  
 descriptors['LogP'] = Chem.Crippen.MolLogP(mol)  
 descriptors['TPSA'] = Chem.rdMolDescriptors.CalcTPSA(mol)  
 return descriptors  
  
# Tải dữ liệu  
data = pd.read\_csv(os.path.join(base\_path, "chembl\_data.csv")) # Thay thế bằng file CSV của bạn  
  
# Xử lý SMILES bị thiếu  
data = data.dropna(subset=['canonical\_smiles'])  
  
# Tính toán các descriptor  
data['descriptors'] = data['canonical\_smiles'].apply(tinh\_toan\_descriptor)  
data = data.dropna(subset=['descriptors'])  
data = data[data['descriptors'].notna()]  
  
# Chuyển descriptor thành các cột  
data = pd.concat([data.drop(['descriptors'], axis=1), data['descriptors'].apply(pd.Series)], axis=1)  
  
  
# Chuyển đổi IC50 sang pIC50  
def ic50\_sang\_pic50(ic50):  
 """Chuyển đổi IC50 sang pIC50."""  
 if pd.isna(ic50):  
 return np.nan  
 ic50 = float(ic50)  
 if ic50 <= 0:  
 return np.nan  
 pIC50 = -np.log10(ic50 / 1e9)  
 return pIC50  
  
data['pIC50'] = data['standard\_value'].apply(ic50\_sang\_pic50)  
  
# Hiển thị một vài dòng đầu tiên  
print(data.head())

**c. Python Code (Model Building)**

# English  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.preprocessing import StandardScaler  
  
# Select features and target  
features = ['Molecular Weight', 'LogP', 'TPSA'] # Add more features as needed  
target = 'pIC50'  
  
# Handle missing values  
data = data.dropna(subset=features + [target])  
  
# Scale features  
scaler = StandardScaler()  
X = scaler.fit\_transform(data[features])  
y = data[target]  
  
# 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 = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Evaluate model  
y\_pred = model.predict(X\_test)  
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}")  
  
# Vietnamese  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.preprocessing import StandardScaler  
  
# Chọn các đặc trưng và mục tiêu  
features = ['Molecular Weight', 'LogP', 'TPSA'] # Thêm các đặc trưng khác nếu cần  
target = 'pIC50'  
  
# Xử lý các giá trị bị thiếu  
data = data.dropna(subset=features + [target])  
  
# Chuẩn hóa các đặc trưng  
scaler = StandardScaler()  
X = scaler.fit\_transform(data[features])  
y = data[target]  
  
# 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 = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Đánh giá mô hình  
y\_pred = model.predict(X\_test)  
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}")

**Giải thích lỗi và cách sửa:**

* **Lỗi:** ValueError:squaredis deprecated and will be removed in scikit-learn 1.4. Usesquared=False` if you want to calculate the root mean squared error.
* **Nguyên nhân:** Phiên bản scikit-learn của bạn cũ và không hỗ trợ tham số squared=False trong hàm mean\_squared\_error.
* **Cách sửa:** Nâng cấp scikit-learn lên phiên bản mới nhất bằng lệnh pip install -U scikit-learn. Nếu không thể nâng cấp, bạn có thể tính RMSE bằng cách lấy căn bậc hai của MSE: rmse = mean\_squared\_error(y\_test, y\_pred, squared=False). Tuy nhiên, tôi khuyến khích bạn nâng cấp scikit-learn để tận dụng các tính năng mới nhất và sửa lỗi.

**4. Ví dụ Code SQL và Python mẫu**

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

**Ví dụ 1: Lọc các hợp chất dựa trên trọng lượng phân tử (Filtering compounds based on molecular weight)**

* **SQL:**

-- English  
SELECT  
 md.molregno,  
 cs.canonical\_smiles  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
WHERE  
 md.mw\_freebase BETWEEN 200 AND 400  
LIMIT 100;  
  
-- Vietnamese  
SELECT  
 md.molregno,  
 cs.canonical\_smiles  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
WHERE  
 md.mw\_freebase BETWEEN 200 AND 400  
LIMIT 100;

* **Python:**

# English  
import pandas as pd  
from rdkit import Chem  
  
# Load data (replace with your actual loading method)  
data = pd.read\_csv("chembl\_data.csv")  
data = data.dropna(subset=['canonical\_smiles'])  
  
# Function to calculate molecular weight using RDKit  
def calculate\_mw(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Chem.Descriptors.MolWt(mol)  
 return None  
  
# Apply the function to calculate molecular weight  
data['MW'] = data['canonical\_smiles'].apply(calculate\_mw)  
  
# Filter compounds with MW between 200 and 400  
filtered\_data = data[(data['MW'] >= 200) & (data['MW'] <= 400)]  
  
print(filtered\_data.head())  
  
  
# Vietnamese  
import pandas as pd  
from rdkit import Chem  
  
# Tải dữ liệu (thay thế bằng phương pháp tải dữ liệu thực tế của bạn)  
data = pd.read\_csv("chembl\_data.csv")  
data = data.dropna(subset=['canonical\_smiles'])  
  
# Hàm tính toán trọng lượng phân tử bằng RDKit  
def tinh\_trong\_luong\_phan\_tu(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return Chem.Descriptors.MolWt(mol)  
 return None  
  
# Áp dụng hàm để tính toán trọng lượng phân tử  
data['MW'] = data['canonical\_smiles'].apply(tinh\_trong\_luong\_phan\_tu)  
  
# Lọc các hợp chất có MW trong khoảng 200 và 400  
filtered\_data = data[(data['MW'] >= 200) & (data['MW'] <= 400)]  
  
print(filtered\_data.head())

**Ví dụ 2: Tính toán số lượng vòng (Calculating the number of rings)**

* **SQL:** (Không thể thực hiện trực tiếp trong SQL, cần sử dụng Python)
* **Python:**

# English  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import rdMolDescriptors  
  
# Load data  
data = pd.read\_csv("chembl\_data.csv")  
data = data.dropna(subset=['canonical\_smiles'])  
  
# Function to calculate the number of rings  
def calculate\_num\_rings(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return rdMolDescriptors.CalcNumRings(mol)  
 return None  
  
# Apply the function  
data['NumRings'] = data['canonical\_smiles'].apply(calculate\_num\_rings)  
  
print(data.head())  
  
  
# Vietnamese  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import rdMolDescriptors  
  
# Tải dữ liệu  
data = pd.read\_csv("chembl\_data.csv")  
data = data.dropna(subset=['canonical\_smiles'])  
  
# Hàm tính toán số lượng vòng  
def tinh\_so\_luong\_vong(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 return rdMolDescriptors.CalcNumRings(mol)  
 return None  
  
# Áp dụng hàm  
data['NumRings'] = data['canonical\_smiles'].apply(tinh\_so\_luong\_vong)  
  
print(data.head())

**Ví dụ 3: Phân tích hoạt tính sinh học (Analyzing activity data)**

* **SQL:**

-- English  
SELECT  
 act.standard\_type,  
 AVG(act.standard\_value) AS average\_value,  
 COUNT(\*) AS num\_measurements  
FROM  
 activities act  
WHERE  
 act.standard\_type IN ('IC50', 'Ki', 'EC50')  
 AND act.standard\_value IS NOT NULL  
GROUP BY  
 act.standard\_type;  
  
-- Vietnamese  
SELECT  
 act.standard\_type,  
 AVG(act.standard\_value) AS average\_value,  
 COUNT(\*) AS num\_measurements  
FROM  
 activities act  
WHERE  
 act.standard\_type IN ('IC50', 'Ki', 'EC50')  
 AND act.standard\_value IS NOT NULL  
GROUP BY  
 act.standard\_type;

* **Python:** (Có thể dùng để trực quan hóa kết quả)

# English  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# Load data (assuming you have activity data in a CSV)  
data = pd.read\_csv("chembl\_data.csv")  
  
# Group by standard\_type and calculate the mean standard\_value  
activity\_summary = data.groupby('standard\_type')['standard\_value'].mean()  
  
# Plot the activity summary  
activity\_summary.plot(kind='bar')  
plt.title('Average Activity Values by Standard Type')  
plt.xlabel('Standard Type')  
plt.ylabel('Average Standard Value')  
plt.show()  
  
  
# Vietnamese  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# Tải dữ liệu (giả sử bạn có dữ liệu hoạt tính trong một file CSV)  
data = pd.read\_csv("chembl\_data.csv")  
  
# Nhóm theo standard\_type và tính giá trị trung bình của standard\_value  
activity\_summary = data.groupby('standard\_type')['standard\_value'].mean()  
  
# Vẽ biểu đồ tóm tắt hoạt tính  
activity\_summary.plot(kind='bar')  
plt.title('Giá trị Hoạt tính Trung bình theo Loại Tiêu chuẩn')  
plt.xlabel('Loại Tiêu chuẩn')  
plt.ylabel('Giá trị Tiêu chuẩn Trung bình')  
plt.show()

**Ví dụ 4: Sử dụng Random Forest để dự đoán hoạt tính (Using Random Forest for activity prediction)**

* **Python:**

# English  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Load data and calculate descriptors (as shown in previous examples)  
# Assuming 'data' DataFrame is already prepared  
  
# Prepare data  
features = ['Molecular Weight', 'LogP', 'TPSA'] # Adjust based on your available descriptors  
target = 'pIC50'  
  
data = data.dropna(subset=features + [target])  
  
X = data[features]  
y = data[target]  
  
# Split data  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Train Random Forest 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)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")  
  
# Vietnamese  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
# Tải dữ liệu và tính toán descriptor (như trong các ví dụ trước)  
# Giả sử DataFrame 'data' đã được chuẩn bị  
  
# Chuẩn bị dữ liệu  
features = ['Molecular Weight', 'LogP', 'TPSA'] # Điều chỉnh dựa trên các descriptor có sẵn  
target = 'pIC50'  
  
data = data.dropna(subset=features + [target])  
  
X = data[features]  
y = data[target]  
  
# 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 Random Forest  
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)  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")

**Ví dụ 5: Tính toán Murcko Fragments (Calculating Murcko Fragments)**

* **Python:**

# English  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import MurckoFragment  
  
# Load data  
data = pd.read\_csv("chembl\_data.csv")  
data = data.dropna(subset=['canonical\_smiles'])  
  
# Function to calculate Murcko Fragment  
def calculate\_murcko(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 core = MurckoFragment.GetMurckoFragment(mol)  
 return Chem.MolToSmiles(core)  
 return None  
  
# Apply the function  
data['MurckoFragment'] = data['canonical\_smiles'].apply(calculate\_murcko)  
  
print(data.head())  
  
# Vietnamese  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import MurckoFragment  
  
# Tải dữ liệu  
data = pd.read\_csv("chembl\_data.csv")  
data = data.dropna(subset=['canonical\_smiles'])  
  
# Hàm tính toán Murcko Fragment  
def tinh\_murcko(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 core = MurckoFragment.GetMurckoFragment(mol)  
 return Chem.MolToSmiles(core)  
 return None  
  
# Áp dụng hàm  
data['MurckoFragment'] = data['canonical\_smiles'].apply(tinh\_murcko)  
  
print(data.head())

**Lưu ý quan trọng:**

* Đảm bảo bạn đã cài đặt tất cả các thư viện cần thiết (RDKit, Pandas, Scikit-learn, Matplotlib, Seaborn). Bạn có thể sử dụng pip install rdkit pandas scikit-learn matplotlib seaborn để cài đặt chúng.
* Điều chỉnh đường dẫn file và tên cột cho phù hợp với dữ liệu của bạn.
* Các ví dụ trên chỉ là điểm khởi đầu. Bạn có thể tùy chỉnh và mở rộng chúng để phù hợp với nhu cầu cụ thể của dự án.

Tôi hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn tiến hành dự án phân tích dữ liệu ChEMBL 35 một cách hiệu quả. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi! Chúc bạn thành công!