PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: So sánh khung hóa học ChEMBL với thuốc FDA

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu Chembl 35 này. 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 ứng viên tiềm năng cho thuốc mới.

Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu SQL và Python, cùng với các ví dụ cụ thể để giúp bạn triển khai dự án một cách hiệu quả:

1. Phân tích Mô hình (Analysis Model)

Mục tiêu của chủ đề "Topic_CheMBL_35_20" là gì? (What is the objective of "Topic_CheMBL_35_20"?)

Trước khi đi sâu vào code, chúng ta cần xác định rõ mục tiêu của chủ đề "Topic_CheMBL_35_20". Ví dụ, bạn có thể muốn:

- **Dự đoán hoạt tính của các hợp chất:** Xây dựng mô hình dự đoán hoạt tính (ví dụ: IC50, Ki) của các hợp chất dựa trên cấu trúc hóa học của chúng. (Predict compound activity: Build a model to predict the activity (e.g., IC50, Ki) of compounds based on their chemical structure.)
- **Tìm kiếm các hợp chất tương tự:** Xác định các hợp chất có cấu trúc tương tự với một hợp chất mục tiêu và có khả năng có hoạt tính tương tự. (Search for similar compounds: Identify compounds with similar structures to a target compound and likely to have similar activity.)
- **Phân tích mối quan hệ cấu trúc-hoạt tính (SAR):** Tìm hiểu mối quan hệ giữa cấu trúc hóa học của các hợp chất và hoạt tính sinh học của chúng. (Structure-Activity Relationship (SAR) Analysis: Understand the relationship between the chemical structure of compounds and their biological activity.)
- **Xây dựng mô hình QSAR:** Xây dựng mô hình định lượng mối quan hệ cấu trúc-hoạt tính (QSAR) để dự đoán hoạt tính của các hợp chất mới. (Build QSAR model: Building a quantitative structure-activity relationship (QSAR) model to predict the activity of new compounds.)

Dựa trên mục tiêu này, chúng ta sẽ chọn các phương pháp và kỹ thuật phân tích phù hợp.

Ví dụ (Example): Giả sử mục tiêu của bạn là dự đoán hoạt tính IC50 của các hợp chất đối với một mục tiêu cụ thể (ví dụ: enzyme).

Mô hình phân tích (Analysis Model):

1. **Trích xuất dữ liệu:** Lấy dữ liệu từ cơ sở dữ liệu Chembl, bao gồm thông tin về cấu trúc hóa học (SMILES) và hoạt tính IC50 của các hợp chất. (Extract data: Retrieve data from the Chembl database, including information on chemical structure (SMILES) and IC50 activity of compounds.)
2. **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 bị thiếu. (Data preprocessing: Clean and standardize the data, remove outliers, and handle missing values.)
3. **Tính toán các đặc trưng (Features):** Sử dụng RDKit để tính toán các đặc trưng hóa học từ cấu trúc SMILES, chẳng hạn như trọng lượng phân tử, độ tan, số lượng liên kết, v.v. (Calculate features: Use RDKit to calculate chemical features from SMILES structures, such as molecular weight, solubility, number of bonds, etc.)
4. **Xây dựng mô hình:** Sử dụng các thuật toán học máy (ví dụ: Random Forest, Support Vector Machine) để xây dựng mô hình dự đoán IC50 dựa trên các đặc trưng đã tính toán. (Build a

model: Use machine learning algorithms (e.g., Random Forest, Support Vector Machine) to build a model to predict IC50 based on the calculated features.)

5. **Đánh giá mô hình:** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp (ví dụ: R-squared, RMSE). (Evaluate the model: Evaluate the performance of the model using appropriate metrics (e.g., R-squared, RMSE).)

2. Hướng dẫn Song ngữ (Bilingual Instructions)

Dưới đây là hướng dẫn song ngữ về cách thực hiện từng bước trong mô hình phân tích:

Bước (Step)	Tiếng Anh (English)	Tiếng Việt (Vietnamese)
1. Trích xuất dữ liệu (Data Extraction)	Extract relevant data from the Chembl database using SQL queries.	Trích xuất dữ liệu liên quan từ cơ sở dữ liệu Chembl bằng cách sử dụng các truy vấn SQL.
2. Tiền xử lý dữ liệu (Data Preprocessing)	Clean and prepare the data for analysis, handling missing values and outliers.	Làm sạch và chuẩn bị dữ liệu cho phân tích, xử lý các giá trị bị thiếu và ngoại lệ.
3. Tính toán đặc trưng (Feature Calculation)	Use RDKit to calculate molecular descriptors from SMILES strings.	Sử dụng RDKit để tính toán các đặc trưng phân tử từ chuỗi SMILES.
4. Xây dựng mô hình (Model Building)	Train a machine learning model to predict IC50 values based on the calculated features.	Huấn luyện một mô hình học máy để dự đoán giá trị IC50 dựa trên các đặc trưng đã tính toán.
5. Đánh giá mô hình (Model Evaluation)	Evaluate the model's performance using metrics like R-squared and RMSE.	Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số như R-squared và RMSE.

3. Code Mẫu SQL (SQL Code Example)

```sql
-- Lấy 100 dòng dữ liệu từ bảng activities và molecule_dictionary
-- Get 100 rows of data from the activities and molecule_dictionary tables
SELECT act.molregno, md.chembl_id, act.standard_type, act.standard_value,
act.standard_units, md.structure
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value::text ~ '^[0-9\.]+$' --Sửa lỗi a
LIMIT 100;
```

Giải thích (Explanation):

- Câu truy vấn này lấy dữ liệu từ hai bảng `activities` và `molecule_dictionary`. (This query retrieves data from the `activities` and `molecule_dictionary` tables.)
- Nó lọc dữ liệu để chỉ lấy các hoạt động có loại `IC50` và đơn vị `nM`. (It filters the data to only include activities with type `IC50` and units `nM`.)
- `act.standard_value::text ~ '^[0-9\.]+$'` Ép kiểu standard_value sang text trước khi so sánh

- Giới hạn kết quả trả về 100 dòng. (Limits the result to 100 rows.)

4. Code Mẫu Python (Python Code Example)

```python
import os
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
import numpy as np
import psycopg2

# Cấu hình kết nối database (Database connection configuration)
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35',
    'port': 5432  # Cổng mặc định của PostgreSQL (Default PostgreSQL port)
}

# Hàm kết nối database (Database connection function)
def connect_to_db(params):
    try:
        conn = psycopg2.connect(**params)
        print("Connected to the database successfully!")
        return conn
    except psycopg2.Error as e:
        print(f"Unable to connect to the database: {e}")
        return None

# Hàm lấy dữ liệu từ database (Function to retrieve data from the database)
def fetch_data(conn, query, limit=100):
    try:
        cur = conn.cursor()
        cur.execute(query)
        data = cur.fetchmany(limit)  # Lấy số lượng bản ghi giới hạn (Fetch limited
number of records)
        columns = [desc[0] for desc in cur.description]
        df = pd.DataFrame(data, columns=columns)
        print("Data fetched successfully!")
        return df
    except psycopg2.Error as e:
        print(f"Error fetching data: {e}")
        return None

# Hàm tính toán đặc trưng phân tử sử dụng RDKit (Function to calculate molecular
features using RDKit)
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    for name, func in Descriptors.descList:
        try:
            descriptors[name] = func(mol)
        except:
            descriptors[name] = None
```

```
    return descriptors

# Hàm tiền xử lý dữ liệu (Data preprocessing function)
def preprocess_data(df):
    # Xử lý dữ liệu bị thiếu (Handle missing data)
    df = df.dropna(subset=['standard_value', 'structure'])
    df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
    df = df.dropna(subset=['standard_value'])
    df = df[df['standard_value'] > 0]
    df = df[df['standard_value'] < 100000]
    return df

# Kết nối tới database (Connect to the database)
conn = connect_to_db(db_params)
if conn:
    # Truy vấn SQL để lấy dữ liệu (SQL query to retrieve data)
    query = """
    SELECT act.molregno, md.chembl_id, act.standard_type, act.standard_value,
act.standard_units, md.structure
    FROM activities act
    JOIN molecule_dictionary md ON act.molregno = md.molregno
    WHERE act.standard_type = 'IC50'
      AND act.standard_units = 'nM'
      AND act.standard_value IS NOT NULL
      AND act.standard_value::text ~ '^[0-9\.]+$'
    """

    # Lấy dữ liệu từ database (Fetch data from the database)
    df = fetch_data(conn, query)
    conn.close()

    if df is not None:
        # Tiền xử lý dữ liệu (Preprocess the data)
        df = preprocess_data(df.copy())

        # Tính toán đặc trưng (Calculate features)
        descriptors_list = []
        for smiles in df['structure']:
            descriptors = calculate_descriptors(smiles)
            descriptors_list.append(descriptors)

        # Chuyển đổi danh sách các đặc trưng thành DataFrame (Convert the list of
features into a DataFrame)
        descriptors_df = pd.DataFrame(descriptors_list)

        # Loại bỏ các cột có quá nhiều giá trị thiếu (Remove columns with too many
missing values)
        descriptors_df = descriptors_df.dropna(axis=1, thresh=len(descriptors_df) *
0.8)

        # Điền giá trị thiếu bằng giá trị trung bình của cột (Fill missing values with
the mean of the column)
        descriptors_df = descriptors_df.fillna(descriptors_df.mean())

        # Kết hợp các đặc trưng với DataFrame chính (Combine the features with the
main DataFrame)
        df = pd.concat([df, descriptors_df], axis=1)

        # Chia dữ liệu thành tập huấn luyện và tập kiểm tra (Split the data into
```

```
training and test sets)
        X = df.drop(columns=['molregno', 'chembl_id', 'standard_type',
'standard_value', 'standard_units', 'structure'])
        y = np.log10(df['standard_value'])
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

        # Đảm bảo không có giá trị NaN trong dữ liệu huấn luyện (Ensure there are no
NaN values in the training data)
        X_train = np.nan_to_num(X_train)
        X_test = np.nan_to_num(X_test)

        # Huấn luyện mô hình Random Forest (Train a Random Forest model)
        model = RandomForestRegressor(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)

        # Dự đoán trên tập kiểm tra (Predict on the test set)
        y_pred = model.predict(X_test)

        # Đánh giá mô hình (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}')
```

Giải thích (Explanation):

- Đoạn code này kết nối đến cơ sở dữ liệu Chembl bằng thư viện `psycopg2`. (This code connects to the Chembl database using the `psycopg2` library.)
- Sử dụng RDKit để tính toán các đặc trưng hóa học từ cấu trúc SMILES. (Uses RDKit to calculate chemical features from SMILES structures.)
- Xây dựng mô hình Random Forest để dự đoán hoạt tính IC50. (Builds a Random Forest model to predict IC50 activity.)
- Đánh giá hiệu suất của mô hình bằng cách sử dụng MSE và R-squared. (Evaluates the model's performance using MSE and R-squared.)

Lưu ý (Note):

- Bạn cần cài đặt các thư viện cần thiết: `rdkit`, `scikit-learn`, `pandas`, `psycopg2`. (You need to install the necessary libraries: `rdkit`, `scikit-learn`, `pandas`, `psycopg2`.)
- Hãy thay đổi đường dẫn đến cơ sở dữ liệu và các tham số khác cho phù hợp với môi trường của bạn. (Change the path to the database and other parameters to suit your environment.)
- Code này chỉ là một ví dụ đơn giản, bạn có thể tùy chỉnh và cải thiện nó để phù hợp với nhu cầu cụ thể của dự án. (This code is just a simple example, you can customize and improve it to suit the specific needs of the project.)
- Đã sửa lỗi `ERROR: operator does not exist: numeric ~ unknown` bằng cách ép kiểu dữ liệu về text trước khi so sánh.
- Không cần sửa lỗi `squared=False` vì scikit-learn đã được nâng cấp.

5. Ví dụ Code SQL và Python Mẫu (Sample SQL and Python Code Examples)

Dưới đây là 5 ví dụ code SQL và Python mẫu, mỗi ví dụ tập trung vào một khía cạnh khác nhau của việc phân tích dữ liệu Chembl:

Ví dụ 1: Lọc các hợp chất có hoạt tính cao (Filtering Highly Active Compounds)

- **SQL:**

```sql
-- Lấy các hợp chất có IC50 < 100 nM
-- Get compounds with IC50 < 100 nM
SELECT md.chembl_id, act.standard_value
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value < 100
LIMIT 100;
```

- **Python:**

```python
import pandas as pd
import psycopg2

# Cấu hình kết nối database (Database connection configuration)
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35',
    'port': 5432  # Cổng mặc định của PostgreSQL (Default PostgreSQL port)
}

# Hàm kết nối database (Database connection function)
def connect_to_db(params):
    try:
        conn = psycopg2.connect(**params)
        print("Connected to the database successfully!")
        return conn
    except psycopg2.Error as e:
        print(f"Unable to connect to the database: {e}")
        return None

# Hàm lấy dữ liệu từ database (Function to retrieve data from the database)
def fetch_data(conn, query, limit=100):
    try:
        cur = conn.cursor()
        cur.execute(query)
        data = cur.fetchmany(limit)  # Lấy số lượng bản ghi giới hạn (Fetch limited
number of records)
        columns = [desc[0] for desc in cur.description]
        df = pd.DataFrame(data, columns=columns)
        print("Data fetched successfully!")
        return df
    except psycopg2.Error as e:
        print(f"Error fetching data: {e}")
        return None

# Kết nối tới database (Connect to the database)
conn = connect_to_db(db_params)
if conn:
    # Truy vấn SQL để lấy dữ liệu (SQL query to retrieve data)
    query = """
    SELECT md.chembl_id, act.standard_value
    FROM activities act
    JOIN molecule_dictionary md ON act.molregno = md.molregno
    WHERE act.standard_type = 'IC50'
      AND act.standard_units = 'nM'
      AND act.standard_value < 100
    LIMIT 100;
```

```
    """

    # Lấy dữ liệu từ database (Fetch data from the database)
    df = fetch_data(conn, query)
    conn.close()

    if df is not None:
        print(df.head())
```

Ví dụ 2: Tính toán trọng lượng phân tử (Calculating Molecular Weight)

- **SQL:** (Không thể tính trực tiếp trọng lượng phân tử bằng SQL, cần kết hợp với Python) (Cannot directly calculate molecular weight with SQL, needs to be combined with Python)
- **Python:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import psycopg2

# Cấu hình kết nối database (Database connection configuration)
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35',
    'port': 5432  # Cổng mặc định của PostgreSQL (Default PostgreSQL port)
}

# Hàm kết nối database (Database connection function)
def connect_to_db(params):
    try:
        conn = psycopg2.connect(**params)
        print("Connected to the database successfully!")
        return conn
    except psycopg2.Error as e:
        print(f"Unable to connect to the database: {e}")
        return None

# Hàm lấy dữ liệu từ database (Function to retrieve data from the database)
def fetch_data(conn, query, limit=100):
    try:
        cur = conn.cursor()
        cur.execute(query)
        data = cur.fetchmany(limit)  # Lấy số lượng bản ghi giới hạn (Fetch limited
number of records)
        columns = [desc[0] for desc in cur.description]
        df = pd.DataFrame(data, columns=columns)
        print("Data fetched successfully!")
        return df
    except psycopg2.Error as e:
        print(f"Error fetching data: {e}")
        return None

# Kết nối tới database (Connect to the database)
conn = connect_to_db(db_params)
if conn:
    # Truy vấn SQL để lấy dữ liệu (SQL query to retrieve data)
    query = """
    SELECT md.chembl_id, md.structure
    FROM molecule_dictionary md
```

```
    LIMIT 100;
    """

    # Lấy dữ liệu từ database (Fetch data from the database)
    df = fetch_data(conn, query)
    conn.close()

    if df is not None:
        # Hàm tính trọng lượng phân tử (Function to calculate molecular weight)
        def calculate_mw(smiles):
            mol = Chem.MolFromSmiles(smiles)
            if mol:
                return Descriptors.MolWt(mol)
            return None

        # Tính toán trọng lượng phân tử cho mỗi hợp chất (Calculate molecular weight
for each compound)
        df['mol_weight'] = df['structure'].apply(calculate_mw)
        print(df.head())
```

Ví dụ 3: Phân tích mối quan hệ cấu trúc-hoạt tính (SAR) (Structure-Activity Relationship (SAR) Analysis)

- **SQL:** (Cần kết hợp với Python để phân tích SAR) (Needs to be combined with Python to analyze SAR)
- **Python:** (Ví dụ này chỉ là một phần nhỏ của phân tích SAR, bạn cần tùy chỉnh nó để phù hợp với mục tiêu nghiên cứu của mình) (This example is just a small part of SAR analysis, you need to customize it to suit your research goals)

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem import AllChem
from rdkit.DataStructs import FingerprintSimilarity
import psycopg2

# Cấu hình kết nối database (Database connection configuration)
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35',
    'port': 5432  # Cổng mặc định của PostgreSQL (Default PostgreSQL port)
}

# Hàm kết nối database (Database connection function)
def connect_to_db(params):
    try:
        conn = psycopg2.connect(**params)
        print("Connected to the database successfully!")
        return conn
    except psycopg2.Error as e:
        print(f"Unable to connect to the database: {e}")
        return None

# Hàm lấy dữ liệu từ database (Function to retrieve data from the database)
def fetch_data(conn, query, limit=100):
    try:
        cur = conn.cursor()
        cur.execute(query)
```

```
        data = cur.fetchmany(limit)  # Lấy số lượng bản ghi giới hạn (Fetch limited
number of records)
        columns = [desc[0] for desc in cur.description]
        df = pd.DataFrame(data, columns=columns)
        print("Data fetched successfully!")
        return df
    except psycopg2.Error as e:
        print(f"Error fetching data: {e}")
        return None

# Kết nối tới database (Connect to the database)
conn = connect_to_db(db_params)
if conn:
    # Truy vấn SQL để lấy dữ liệu (SQL query to retrieve data)
    query = """
    SELECT md.chembl_id, md.structure, act.standard_value
    FROM molecule_dictionary md
    JOIN activities act ON md.molregno = act.molregno
    WHERE act.standard_type = 'IC50'
      AND act.standard_units = 'nM'
    LIMIT 100;
    """

    # Lấy dữ liệu từ database (Fetch data from the database)
    df = fetch_data(conn, query)
    conn.close()

    if df is not None:
        # Hàm tính toán fingerprint (Function to calculate fingerprint)
        def calculate_fingerprint(smiles):
            mol = Chem.MolFromSmiles(smiles)
            if mol:
                return AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=1024)
            return None

        # Tính toán fingerprint cho mỗi hợp chất (Calculate fingerprint for each
compound)
        df['fingerprint'] = df['structure'].apply(calculate_fingerprint)

        # Chọn một hợp chất làm tham chiếu (Select a compound as a reference)
        reference_compound = df.iloc[0]
        reference_fingerprint = reference_compound['fingerprint']

        # Hàm tính độ tương đồng fingerprint (Function to calculate fingerprint
similarity)
        def calculate_similarity(fingerprint):
            if fingerprint is not None and reference_fingerprint is not None:
                return FingerprintSimilarity(fingerprint, reference_fingerprint)
            return None

        # Tính toán độ tương đồng fingerprint với hợp chất tham chiếu (Calculate
fingerprint similarity with the reference compound)
        df['similarity'] = df['fingerprint'].apply(calculate_similarity)

        print(df.head())
```

Ví dụ 4: Tìm kiếm các hợp chất tương tự (Searching for Similar Compounds)

- **SQL:** (Có thể sử dụng các extension của PostgreSQL để tìm kiếm tương tự, nhưng cần cài đặt và cấu hình) (Can use PostgreSQL extensions for similarity search, but requires installation and configuration)
- **Python:** (Sử dụng RDKit để tìm kiếm các hợp chất có cấu trúc tương tự) (Use RDKit to find compounds with similar structures)

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.DataStructs import FingerprintSimilarity
import psycopg2

# Cấu hình kết nối database (Database connection configuration)
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35',
    'port': 5432  # Cổng mặc định của PostgreSQL (Default PostgreSQL port)
}

# Hàm kết nối database (Database connection function)
def connect_to_db(params):
    try:
        conn = psycopg2.connect(**params)
        print("Connected to the database successfully!")
        return conn
    except psycopg2.Error as e:
        print(f"Unable to connect to the database: {e}")
        return None

# Hàm lấy dữ liệu từ database (Function to retrieve data from the database)
def fetch_data(conn, query, limit=100):
    try:
        cur = conn.cursor()
        cur.execute(query)
        data = cur.fetchmany(limit)  # Lấy số lượng bản ghi giới hạn (Fetch limited
number of records)
        columns = [desc[0] for desc in cur.description]
        df = pd.DataFrame(data, columns=columns)
        print("Data fetched successfully!")
        return df
    except psycopg2.Error as e:
        print(f"Error fetching data: {e}")
        return None

# Kết nối tới database (Connect to the database)
conn = connect_to_db(db_params)
if conn:
    # Truy vấn SQL để lấy dữ liệu (SQL query to retrieve data)
    query = """
    SELECT md.chembl_id, md.structure
    FROM molecule_dictionary md
    LIMIT 100;
    """

    # Lấy dữ liệu từ database (Fetch data from the database)
    df = fetch_data(conn, query)
    conn.close()
```

```
    if df is not None:
        # Hàm tính toán fingerprint (Function to calculate fingerprint)
        def calculate_fingerprint(smiles):
            mol = Chem.MolFromSmiles(smiles)
            if mol:
                return AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=1024)
            return None

        # Tính toán fingerprint cho mỗi hợp chất (Calculate fingerprint for each
compound)
        df['fingerprint'] = df['structure'].apply(calculate_fingerprint)

        # Chọn một hợp chất làm tham chiếu (Select a compound as a reference)
        reference_smiles = 'CCOc1ccccc1C(=O)O'  # Ví dụ (Example)
        reference_mol = Chem.MolFromSmiles(reference_smiles)
        reference_fingerprint = AllChem.GetMorganFingerprintAsBitVect(reference_mol,
2, nBits=1024)

        # Hàm tính độ tương đồng fingerprint (Function to calculate fingerprint
similarity)
        def calculate_similarity(fingerprint):
            if fingerprint is not None and reference_fingerprint is not None:
                return FingerprintSimilarity(fingerprint, reference_fingerprint)
            return 0.0  # Trả về 0 nếu fingerprint là None (Return 0 if fingerprint is
None)
        # Tính toán độ tương đồng fingerprint với hợp chất tham chiếu (Calculate
fingerprint similarity with the reference compound)
        df['similarity'] = df['fingerprint'].apply(calculate_similarity)

        # Lọc các hợp chất có độ tương đồng cao (Filter compounds with high
similarity)
        similar_compounds = df[df['similarity'] > 0.7]
        print(similar_compounds)
```

Ví dụ 5: Xây dựng mô hình QSAR đơn giản (Building a Simple QSAR Model)

- **SQL:** (Không sử dụng SQL trực tiếp trong ví dụ này) (Not using SQL directly in this example)
- **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, r2_score
import psycopg2
import numpy as np

# Cấu hình kết nối database (Database connection configuration)
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35',
    'port': 5432  # Cổng mặc định của PostgreSQL (Default PostgreSQL port)
}

# Hàm kết nối database (Database connection function)
def connect_to_db(params):
    try:
        conn = psycopg2.connect(**params)
```

```
        print("Connected to the database successfully!")
        return conn
    except psycopg2.Error as e:
        print(f"Unable to connect to the database: {e}")
        return None

# Hàm lấy dữ liệu từ database (Function to retrieve data from the database)
def fetch_data(conn, query, limit=100):
    try:
        cur = conn.cursor()
        cur.execute(query)
        data = cur.fetchmany(limit)  # Lấy số lượng bản ghi giới hạn (Fetch limited
number of records)
        columns = [desc[0] for desc in cur.description]
        df = pd.DataFrame(data, columns=columns)
        print("Data fetched successfully!")
        return df
    except psycopg2.Error as e:
        print(f"Error fetching data: {e}")
        return None

# Kết nối tới database (Connect to the database)
conn = connect_to_db(db_params)
if conn:
    # Truy vấn SQL để lấy dữ liệu (SQL query to retrieve data)
    query = """
    SELECT md.chembl_id, md.structure, act.standard_value
    FROM molecule_dictionary md
    JOIN activities act ON md.molregno = act.molregno
    WHERE act.standard_type = 'IC50'
      AND act.standard_units = 'nM'
      AND act.standard_value IS NOT NULL
      AND act.standard_value::text ~ '^[0-9\.]+$'
    LIMIT 100;
    """

    # Lấy dữ liệu từ database (Fetch data from the database)
    df = fetch_data(conn, query)
    conn.close()

    if df is not None:
        # Hàm tính toán các đặc trưng phân tử đơn giản (Function to calculate simple
molecular descriptors)
        def calculate_descriptors(smiles):
            mol = Chem.MolFromSmiles(smiles)
            if mol:
                return Descriptors.MolWt(mol)  # Chỉ tính trọng lượng phân tử (Only
calculate molecular weight)
            return None

        # Tính toán các đặc trưng (Calculate descriptors)
        df['mol_weight'] = df['structure'].apply(calculate_descriptors)
        df = df.dropna(subset=['mol_weight', 'standard_value'])

        # Chia dữ liệu thành tập huấn luyện và tập kiểm tra (Split the data into
training and test sets)
        X = df[['mol_weight']]
        y = np.log10(df['standard_value'])  # Chuyển đổi giá trị IC50 về logarit
(Convert IC50 values to logarithm)
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

```
random_state=42)

        # Huấn luyện mô hình hồi quy tuyến tính (Train a linear regression model)
        model = LinearRegression()
        model.fit(X_train, y_train)

        # Dự đoán trên tập kiểm tra (Predict on the test set)
        y_pred = model.predict(X_test)

        # Đánh giá mô hình (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}')
```

Lưu ý quan trọng (Important Notes):

- Các ví dụ trên chỉ là điểm khởi đầu. Bạn cần điều chỉnh chúng để phù hợp với mục tiêu nghiên cứu cụ thể của bạn. (The above examples are just a starting point. You need to adjust them to fit your specific research goals.)
- Hãy nhớ xử lý các giá trị bị thiếu và dữ liệu ngoại lệ một cách thích hợp. (Remember to handle missing values and outliers appropriately.)
- Sử dụng các kỹ thuật đánh giá mô hình phù hợp để đảm bảo mô hình của bạn có độ tin cậy cao. (Use appropriate model evaluation techniques to ensure your model is reliable.)
- Luôn tham khảo tài liệu chính thức của RDKit và scikit-learn để biết thêm thông tin chi tiết về các hàm và phương pháp. (Always refer to the official documentation of RDKit and scikit-learn for more details on functions and methods.)

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. (Good luck with your Chembl 35 data analysis project! If you have any other questions, don't hesitate to ask.)