PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Thiết kế thư viện hóa học tập trung từ ChEMBL

Tuyệt vời! Tôi sẽ hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit này. Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu SQL và Python, cùng với các lưu ý quan trọng để giải quyết các lỗi bạn gặp phải.

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

Dự án của bạn tập trung vào việc khai thác dữ liệu từ cơ sở dữ liệu ChEMBL 35 để hỗ trợ nghiên cứu và phát triển thuốc. Việc kết hợp ChEMBL với RDKit cho phép bạn tận dụng thông tin về hoạt tính sinh học của các hợp chất và cấu trúc hóa học của chúng.

Mô hình phân tích bao gồm các bước chính sau:

- **Trích xuất dữ liệu (Data Extraction):** Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35, tập trung vào các thông tin liên quan đến hoạt tính sinh học (ví dụ: IC50, Ki) và cấu trúc hóa học (ví dụ: SMILES).
- **Tiền xử lý dữ liệu (Data Preprocessing):** Làm sạch và chuẩn hóa dữ liệu, bao gồm xử lý các giá trị thiếu, loại bỏ dữ liệu không hợp lệ và chuyển đổi dữ liệu về định dạng phù hợp cho phân tích.
- **Tính toán descriptor (Descriptor Calculation):** Sử dụng RDKit để tính toán các descriptor hóa học từ cấu trúc SMILES của các hợp chất. Các descriptor này có thể bao gồm các thuộc tính vật lý, hóa học và cấu trúc của phân tử.
- **Phân tích thống kê và mô hình hóa (Statistical Analysis and Modeling):** Sử dụng các phương pháp thống kê và học máy để phân tích mối quan hệ giữa các descriptor hóa học và hoạt tính sinh học. Điều này có thể bao gồm xây dựng các mô hình dự đoán hoạt tính (ví dụ: QSAR/QSPR), phân cụm các hợp chất dựa trên đặc điểm hóa học, hoặc xác định các đặc điểm quan trọng liên quan đến hoạt tính.
- **Trực quan hóa dữ liệu (Data Visualization):** Sử dụng các công cụ trực quan hóa để khám phá dữ liệu và trình bày kết quả phân tích một cách rõ ràng và dễ hiểu.

2. Hướng dẫn song ngữ (Bilingual Instructions)

English:

This project aims to analyze data from the ChEMBL 35 database to support drug discovery and development. By combining ChEMBL with RDKit, you can leverage information on the biological activity of compounds and their chemical structures.

Key steps include:

- **Data Extraction:** Use SQL to query data from the ChEMBL 35 database, focusing on information related to biological activity (e.g., IC50, Ki) and chemical structures (e.g., SMILES).
- **Data Preprocessing:** Clean and standardize the data, including handling missing values, removing invalid data, and converting data to a suitable format for analysis.
- **Descriptor Calculation:** Use RDKit to calculate chemical descriptors from the SMILES structures of the compounds. These descriptors can include physical, chemical, and structural properties of the molecule.
- **Statistical Analysis and Modeling:** Use statistical and machine learning methods to analyze the relationship between chemical descriptors and biological activity. This can include building

predictive activity models (e.g., QSAR/QSPR), clustering compounds based on chemical characteristics, or identifying important features related to activity.
- **Data Visualization:** Use visualization tools to explore the data and present the analysis results in a clear and understandable manner.

Tiếng Việt:

Dự án này nhằm phân tích dữ liệu từ cơ sở dữ liệu ChEMBL 35 để hỗ trợ nghiên cứu và phát triển thuốc. Bằng cách kết hợp ChEMBL với RDKit, bạn có thể tận dụng thông tin về hoạt tính sinh học của các hợp chất và cấu trúc hóa học của chúng.

Các bước chính bao gồm:

- **Trích xuất dữ liệu:** Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35, tập trung vào thông tin liên quan đến hoạt tính sinh học (ví dụ: IC50, Ki) và cấu trúc hóa học (ví dụ: SMILES).
- **Tiền xử lý dữ liệu:** Làm sạch và chuẩn hóa dữ liệu, bao gồm xử lý các giá trị thiếu, loại bỏ dữ liệu không hợp lệ và chuyển đổi dữ liệu về định dạng phù hợp cho phân tích.
- **Tính toán descriptor:** Sử dụng RDKit để tính toán các descriptor hóa học từ cấu trúc SMILES của các hợp chất. Các descriptor này có thể bao gồm các thuộc tính vật lý, hóa học và cấu trúc của phân tử.
- **Phân tích thống kê và mô hình hóa:** Sử dụng các phương pháp thống kê và học máy để phân tích mối quan hệ giữa các descriptor hóa học và hoạt tính sinh học. Điều này có thể bao gồm xây dựng các mô hình dự đoán hoạt tính (ví dụ: QSAR/QSPR), phân cụm các hợp chất dựa trên đặc điểm hóa học, hoặc xác định các đặc điểm quan trọng liên quan đến hoạt tính.
- **Trực quan hóa dữ liệu:** Sử dụng các công cụ trực quan hóa để khám phá dữ liệu và trình bày kết quả phân tích một cách rõ ràng và dễ hiểu.

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

a. SQL (English & Vietnamese)

```sql
-- English: SQL query to extract data from ChEMBL 35 (limited to 100 rows)
-- Vietnamese: Truy vấn SQL để trích xuất dữ liệu từ ChEMBL 35 (giới hạn 100 dòng)

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.standard_type = 'IC50' -- You can change this to other activity types
    AND act.standard_units = 'nM'
    --AND act.standard_value ~ '^[0-9\.]+$' --Fixing the error, remove this.
LIMIT 100;
```

Lý giải lỗi SQL:

Lỗi `ERROR: operator does not exist: numeric ~ unknown` xảy ra vì bạn đang cố gắng sử dụng toán tử `~` (regular expression match) trên một cột số (`act.standard_value`). Thay vì sử dụng regular expression, bạn có thể bỏ qua dòng này nếu bạn chỉ muốn lọc các giá trị số. Nếu bạn vẫn muốn lọc theo regular expression, bạn cần đảm bảo rằng cột `act.standard_value` được ép kiểu về kiểu text

trước khi sử dụng toán tử ~. Tuy nhiên, trong trường hợp này, có vẻ như việc bỏ qua dòng này là giải pháp đơn giản nhất.

b. Python (English & Vietnamese)

```python
# English: Python code to read data from CSV, calculate descriptors, and perform basic analysis
# Vietnamese: Mã Python để đọc dữ liệu từ CSV, tính toán descriptor và thực hiện phân tích cơ bản

import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import os

# Define base path
base_path = "../data"  # Adjust this to your actual base path

# Construct the file path
csv_file_path = os.path.join(base_path, "your_extracted_data.csv") # Replace "your_extracted_data.csv" with your actual file name

# Read the CSV file
try:
    df = pd.read_csv(csv_file_path)
    print("CSV file loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
    exit()

# Function to calculate molecular weight
def calculate_mw(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol)
        else:
            return None
    except:
        return None

# Apply the function to the 'canonical_smiles' column
df['molecular_weight'] = df['canonical_smiles'].apply(calculate_mw)

# Print some basic statistics
print(df.describe())

# You can add more analysis here, such as plotting or more advanced calculations
```

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

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

Ví dụ 1: SQL - Lọc theo khoảng giá trị hoạt tính (Filtering by activity range)

```sql
-- English: SQL query to filter compounds with IC50 values between 10 and 100 nM
-- Vietnamese: Truy vấn SQL để lọc các hợp chất có giá trị IC50 nằm trong khoảng từ 10 đến 100 nM

SELECT
    md.chembl_id,
```

```
    cs.canonical_smiles,
    act.standard_value
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'
    AND act.standard_units = 'nM'
    AND act.standard_value >= 10
    AND act.standard_value <= 100
LIMIT 100;
```

Ví dụ 1: Python - Tính toán logP (Calculating logP)

```
# English: Python code to calculate LogP using RDKit
# Vietnamese: Mã Python để tính toán LogP sử dụng RDKit

import pandas as pd
from rdkit import Chem
from rdkit.Chem import Crippen
import os

# Define base path
base_path = "../data"  # Adjust this to your actual base path

# Construct the file path
csv_file_path = os.path.join(base_path, "your_extracted_data.csv") # Replace
"your_extracted_data.csv" with your actual file name

# Read the CSV file
try:
    df = pd.read_csv(csv_file_path)
    print("CSV file loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
    exit()

def calculate_logp(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Crippen.MolLogP(mol)
        else:
            return None
    except:
        return None

df['logp'] = df['canonical_smiles'].apply(calculate_logp)

print(df[['chembl_id', 'canonical_smiles', 'logp']].head())
```

Ví dụ 2: SQL - Lọc theo mục tiêu (Filtering by Target)

```
-- English: SQL query to filter compounds targeting a specific protein (replace
'CHEMBL205' with the actual target ID)
-- Vietnamese: Truy vấn SQL để lọc các hợp chất nhắm mục tiêu vào một protein cụ thể
(thay thế 'CHEMBL205' bằng ID mục tiêu thực tế)
```

```sql
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_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.chembl_id = 'CHEMBL205'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
LIMIT 100;
```

Ví dụ 2: Python - Tính toán số lượng vòng (Calculating Number of Rings)

```python
# English: Python code to calculate the number of rings in each molecule
# Vietnamese: Mã Python để tính toán số lượng vòng trong mỗi phân tử

import pandas as pd
from rdkit import Chem
from rdkit.Chem import rdMolDescriptors
import os

# Define base path
base_path = "../data"  # Adjust this to your actual base path

# Construct the file path
csv_file_path = os.path.join(base_path, "your_extracted_data.csv") # Replace
"your_extracted_data.csv" with your actual file name

# Read the CSV file
try:
    df = pd.read_csv(csv_file_path)
    print("CSV file loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
    exit()

def calculate_num_rings(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return rdMolDescriptors.CalcNumRings(mol)
        else:
            return None
    except:
        return None

df['num_rings'] = df['canonical_smiles'].apply(calculate_num_rings)

print(df[['chembl_id', 'canonical_smiles', 'num_rings']].head())
```

Ví dụ 3: SQL - Lọc theo loại hoạt tính (Filtering by activity type khác IC50)

```sql
-- English: SQL query to filter compounds with Ki values
-- Vietnamese: Truy vấn SQL để lọc các hợp chất có giá trị Ki
```

```sql
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value
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 = 'Ki'
    AND act.standard_units = 'nM'
LIMIT 100;
```

Ví dụ 3: Python - Tính toán chỉ số Topological Polar Surface Area (TPSA)

```python
# English: Python code to calculate TPSA using RDKit
# Vietnamese: Mã Python để tính toán TPSA sử dụng RDKit

import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import os

# Define base path
base_path = "../data"  # Adjust this to your actual base path

# Construct the file path
csv_file_path = os.path.join(base_path, "your_extracted_data.csv") # Replace
"your_extracted_data.csv" with your actual file name

# Read the CSV file
try:
    df = pd.read_csv(csv_file_path)
    print("CSV file loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
    exit()

def calculate_tpsa(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.TPSA(mol)
        else:
            return None
    except:
        return None

df['tpsa'] = df['canonical_smiles'].apply(calculate_tpsa)

print(df[['chembl_id', 'canonical_smiles', 'tpsa']].head())
```

Ví dụ 4: SQL - Kết hợp dữ liệu từ nhiều bảng (Joining data from multiple tables)

```sql
-- English: SQL query to join molecule_dictionary, compound_structures, activities,
and target_dictionary tables
-- Vietnamese: Truy vấn SQL để kết hợp các bảng molecule_dictionary,
compound_structures, activities và target_dictionary
```

```sql
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    td.pref_name AS target_name
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
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
LIMIT 100;
```

Ví dụ 4: Python - Phân tích thống kê mô tả (Descriptive statistics analysis)

```python
# English: Python code for descriptive statistics analysis of activity data
# Vietnamese: Mã Python để phân tích thống kê mô tả dữ liệu hoạt tính

import pandas as pd
import os

# Define base path
base_path = "../data"  # Adjust this to your actual base path

# Construct the file path
csv_file_path = os.path.join(base_path, "your_extracted_data.csv") # Replace
"your_extracted_data.csv" with your actual file name

# Read the CSV file
try:
    df = pd.read_csv(csv_file_path)
    print("CSV file loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
    exit()

# Convert standard_value to numeric, errors='coerce' will replace invalid parsing into
NaN
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')

# Remove rows where 'standard_value' is NaN
df = df.dropna(subset=['standard_value'])

# Perform descriptive statistics
print(df['standard_value'].describe())

# Group by standard_type and calculate mean
print(df.groupby('standard_type')['standard_value'].mean())
```

Ví dụ 5: SQL - Tính số lượng hợp chất cho mỗi mục tiêu (Counting compounds per target)

```sql
-- English: SQL query to count the number of compounds for each target
-- Vietnamese: Truy vấn SQL để đếm số lượng hợp chất cho mỗi mục tiêu
```

```sql
SELECT
    td.chembl_id AS target_chembl_id,
    td.pref_name AS target_name,
    COUNT(DISTINCT md.molregno) AS num_compounds
FROM
    target_dictionary td
JOIN
    activities act ON td.tid = act.tid
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
GROUP BY
    td.chembl_id, td.pref_name
ORDER BY
    num_compounds DESC
LIMIT 100;
```

Ví dụ 5: Python - Trực quan hóa dữ liệu (Data visualization)

```python
# English: Python code to visualize the distribution of molecular weights
# Vietnamese: Mã Python để trực quan hóa sự phân bố của trọng lượng phân tử

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Define base path
base_path = "../data"  # Adjust this to your actual base path

# Construct the file path
csv_file_path = os.path.join(base_path, "your_extracted_data.csv") # Replace
"your_extracted_data.csv" with your actual file name

# Read the CSV file
try:
    df = pd.read_csv(csv_file_path)
    print("CSV file loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
    exit()

# Calculate molecular weight (if not already calculated)
from rdkit import Chem
from rdkit.Chem import Descriptors

def calculate_mw(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol)
        else:
            return None
    except:
        return None

if 'molecular_weight' not in df.columns:
```

```
    df['molecular_weight'] = df['canonical_smiles'].apply(calculate_mw)

# Drop NaN values in 'molecular_weight' column
df = df.dropna(subset=['molecular_weight'])

# Create a histogram
plt.figure(figsize=(10, 6))
sns.histplot(df['molecular_weight'], bins=30, kde=True)
plt.title('Distribution of Molecular Weights')
plt.xlabel('Molecular Weight')
plt.ylabel('Frequency')
plt.show()
```

5. Lưu ý về lỗi (Error Notes)

- **Lỗi SQL:** Như đã giải thích ở trên, bạn có thể loại bỏ điều kiện `act.standard_value ~ '^[0-9\.]+$'` hoặc ép kiểu cột `act.standard_value` về kiểu text trước khi sử dụng toán tử `~`.
- **Lỗi Scikit-learn:** Nếu bạn đang sử dụng phiên bản Scikit-learn cũ, hãy cân nhắc nâng cấp lên phiên bản mới hơn hoặc sử dụng các giải pháp thay thế cho tham số `squared=False`. Ví dụ: bạn có thể tính căn bậc hai của Mean Squared Error (MSE) để có được Root Mean Squared Error (RMSE).

Ví dụ về cách tính RMSE nếu không có `squared=False`:

```
from sklearn.metrics import mean_squared_error
import numpy as np

# Example data
y_true = [3, -0.5, 2, 7]
y_predicted = [2.5, 0.0, 2, 8]

# Calculate MSE
mse = mean_squared_error(y_true, y_predicted)

# Calculate RMSE
rmse = np.sqrt(mse)

print("MSE:", mse)
print("RMSE:", rmse)
```

Lời khuyên:

- **Chia nhỏ vấn đề:** Bắt đầu với các truy vấn SQL và đoạn code Python đơn giản, sau đó dần dần tăng độ phức tạp.
- **Kiểm tra dữ liệu:** Luôn kiểm tra dữ liệu của bạn sau mỗi bước để đảm bảo rằng nó đúng và hợp lệ.
- **Tài liệu hóa:** Ghi lại các bước bạn thực hiện và các quyết định bạn đưa ra để giúp bạn và người khác hiểu được quy trình làm việc của bạn.
- **Sử dụng môi trường ảo:** Tạo một môi trường ảo để quản lý các gói Python của bạn và tránh xung đột.
- **Phiên bản hóa mã:** Sử dụng Git để theo dõi các thay đổi của bạn và cộng tác với người khác.

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