PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Khả năng truy xuất cấu trúc 3D của target từ PDB

Tuyệt vời! Với vai trò là một chuyên gia nghiên cứu và phát triển dược học, tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit. Dựa trên thông tin bạn cung cấp, tôi sẽ tập trung vào việc:

1. **Phân tích mô hình phân tích:** Đề xuất một số hướng phân tích phù hợp với dữ liệu ChEMBL và mục tiêu nghiên cứu phát triển thuốc của bạn.
2. **Hướng dẫn song ngữ:** Cung cấp hướng dẫn chi tiết bằng cả tiếng Anh và tiếng Việt, kèm theo giải thích rõ ràng.
3. **Code mẫu:** Tạo các đoạn code SQL và Python mẫu để bạn có thể bắt đầu và tham khảo.
4. **Xử lý lỗi:** Đưa ra giải pháp cho các lỗi bạn đã gặp phải.
5. **Giới hạn dữ liệu:** Chỉ sử dụng 100 dòng dữ liệu để đảm bảo hiệu suất.

1. Phân tích mô hình phân tích (Analysis Model):

Dựa trên dữ liệu ChEMBL, bạn có thể thực hiện nhiều phân tích khác nhau để hỗ trợ quá trình phát triển thuốc. Dưới đây là một số gợi ý:

- **Phân tích mối quan hệ cấu trúc-hoạt tính (Structure-Activity Relationship - SAR):**
    - **Mục tiêu:** Tìm hiểu mối liên hệ giữa cấu trúc hóa học của các phân tử và hoạt tính sinh học của chúng.
    - **Phương pháp:**
        - Sử dụng RDKit để tính toán các descriptor phân tử (ví dụ: trọng lượng phân tử, logP, số lượng vòng).
        - Sử dụng các mô hình hồi quy hoặc phân loại (ví dụ: hồi quy tuyến tính, random forest, SVM) để dự đoán hoạt tính dựa trên các descriptor.
    - **Ứng dụng:** Xác định các nhóm thế (substituents) quan trọng cho hoạt tính, từ đó thiết kế các phân tử mới có hoạt tính tốt hơn.
- **Phân tích tương quan định lượng cấu trúc-hoạt tính (Quantitative Structure-Activity Relationship - QSAR):**
    - **Mục tiêu:** Xây dựng các mô hình toán học để dự đoán hoạt tính của các phân tử dựa trên cấu trúc của chúng.
    - **Phương pháp:** Tương tự như SAR, nhưng tập trung vào việc xây dựng các mô hình có khả năng dự đoán định lượng (ví dụ: IC50, Ki).
    - **Ứng dụng:** Sàng lọc ảo (virtual screening) để tìm kiếm các phân tử tiềm năng từ các thư viện hóa học lớn.
- **Phân tích cụm (Clustering):**
    - **Mục tiêu:** Phân nhóm các phân tử có cấu trúc hoặc hoạt tính tương tự nhau.
    - **Phương pháp:** Sử dụng các thuật toán phân cụm (ví dụ: k-means, hierarchical clustering) dựa trên các descriptor phân tử hoặc fingerprint.
    - **Ứng dụng:** Xác định các scaffold (khung cấu trúc) phổ biến, tìm kiếm các phân tử tương tự với một chất dẫn đầu (lead compound).
- **Phân tích đa dạng hóa (Diversity Analysis):**
    - **Mục tiêu:** Chọn một tập hợp các phân tử đa dạng từ một thư viện hóa học lớn.

    - **Phương pháp:** Sử dụng các thuật toán chọn mẫu dựa trên khoảng cách cấu trúc hoặc fingerprint.
    - **Ứng dụng:** Xây dựng các thư viện sàng lọc (screening libraries) đa dạng, đảm bảo bao phủ không gian hóa học rộng.
- **Phân tích liên kết (Linkage Analysis):**
    - **Mục tiêu:** Tìm kiếm các mối liên kết giữa các mục tiêu thuốc, các bệnh, và các phân tử.
    - **Phương pháp:** Sử dụng các kỹ thuật khai thác dữ liệu (data mining) trên dữ liệu ChEMBL và các nguồn dữ liệu khác.
    - **Ứng dụng:** Xác định các mục tiêu thuốc tiềm năng cho một bệnh cụ thể, tìm kiếm các phân tử có thể tác động đến nhiều mục tiêu khác nhau.

2. Hướng dẫn song ngữ (Bilingual Guide):

English:

This section provides a bilingual guide (English and Vietnamese) on how to perform data analysis using ChEMBL 35 data with RDKit, focusing on the analysis models mentioned above.

Vietnamese:

Phần này cung cấp hướng dẫn song ngữ (tiếng Anh và tiếng Việt) về cách thực hiện phân tích dữ liệu sử dụng dữ liệu ChEMBL 35 với RDKit, tập trung vào các mô hình phân tích đã đề cập ở trên.

Example: Structure-Activity Relationship (SAR) Analysis

English:

1. **Data Extraction:** Extract relevant data from the ChEMBL database, including compound structures (SMILES) and activity values (e.g., IC50).
2. **Molecular Descriptor Calculation:** Use RDKit to calculate molecular descriptors from the SMILES strings.
3. **Model Building:** Build a regression or classification model to predict activity based on the descriptors.
4. **Analysis:** Analyze the model to identify important descriptors and SAR trends.

Vietnamese:

1. **Trích xuất dữ liệu:** Trích xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL, bao gồm cấu trúc hợp chất (SMILES) và giá trị hoạt tính (ví dụ: IC50).
2. **Tính toán descriptor phân tử:** Sử dụng RDKit để tính toán các descriptor phân tử từ chuỗi SMILES.
3. **Xây dựng mô hình:** Xây dựng mô hình hồi quy hoặc phân loại để dự đoán hoạt tính dựa trên các descriptor.
4. **Phân tích:** Phân tích mô hình để xác định các descriptor quan trọng và xu hướng SAR.

3. Code mẫu (Code Examples):

Dưới đây là 5 ví dụ code SQL và Python mẫu, tập trung vào việc trích xuất dữ liệu, tính toán descriptor và xây dựng mô hình đơn giản.

SQL Examples:

```sql
-- Example 1: Select 100 compounds with IC50 values
SELECT  md.chembl_id,
        cs.canonical_smiles,
        act.standard_value,
        act.standard_units
FROM    molecule_dictionary md
JOIN    compound_structures cs ON md.molregno = cs.molregno
```

```sql
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
LIMIT 100;

-- Example 2: Count the number of compounds for each target
SELECT  td.chembl_id,
        COUNT(DISTINCT md.molregno) AS compound_count
FROM    target_dictionary td
JOIN    component_sequences cs ON td.tid = cs.tid
JOIN    target_components tc ON cs.component_id = tc.component_id
JOIN    component_molecules cm ON tc.component_id = cm.component_id
JOIN    molecule_dictionary md ON cm.molregno = md.molregno
GROUP BY td.chembl_id
ORDER BY compound_count DESC
LIMIT 10;

-- Example 3: Find compounds with activity against a specific target
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' -- Replace with your target ChEMBL ID
  AND   act.standard_type = 'IC50'
  AND   act.standard_units = 'nM'
LIMIT 10;

-- Example 4: Compounds with specific substructure (using SMARTS)
SELECT  md.chembl_id,
        cs.canonical_smiles
FROM    molecule_dictionary md
JOIN    compound_structures cs ON md.molregno = cs.molregno
WHERE   cs.canonical_smiles LIKE '%[c1ccccc1]%' -- Benzene ring
LIMIT 10;

-- Example 5: Average molecular weight of compounds
SELECT AVG(MW)
FROM (
    SELECT  md.chembl_id,
            cs.canonical_smiles,
            mol_weight(cs.canonical_smiles) AS MW
    FROM    molecule_dictionary md
    JOIN    compound_structures cs ON md.molregno = cs.molregno
    LIMIT 100
) AS subquery;
```

Python Examples:

```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.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
```

```
# Define base path (adjust as needed)
base_path = "../data"

# Example 1: Load data from CSV (replace with your actual file path)
csv_file = os.path.join(base_path, "chembl_data.csv")  # Replace with your CSV file
try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}")
    df = pd.DataFrame()  # Create an empty DataFrame to avoid errors
except Exception as e:
    print(f"Error reading CSV: {e}")
    df = pd.DataFrame()

if not df.empty:
    # Ensure SMILES and standard_value columns exist
    if 'canonical_smiles' not in df.columns or 'standard_value' not in df.columns:
        print("Error: 'canonical_smiles' or 'standard_value' column not found in
CSV.")
        df = pd.DataFrame() # Create an empty DataFrame to avoid errors
    else:
        df = df.head(100)  # Limit to 100 rows
        print(f"Loaded {len(df)} rows from {csv_file}")
else:
    print("No data loaded.  Please check the CSV file and its contents.")


def calculate_molecular_weight(smiles):
    """Calculates the molecular weight of a molecule given its SMILES string."""
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol)
        else:
            return np.nan  # Handle invalid SMILES
    except:
        return np.nan  # Handle any other errors

if not df.empty:
    # Example 2: Calculate molecular weight using RDKit
    df['molecular_weight'] = df['canonical_smiles'].apply(calculate_molecular_weight)
    print(df[['canonical_smiles', 'molecular_weight']].head())

    # Example 3: Prepare data for a simple linear regression model (if data is
available)
    df = df.dropna(subset=['standard_value', 'molecular_weight', 'canonical_smiles'])
# Drop rows with missing values

    # Convert standard_value to numeric
    df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
    df = df.dropna(subset=['standard_value'])  # Drop rows with non-numeric
standard_value

    if not df.empty and len(df) > 10:

        X = df[['molecular_weight']]
        y = df['standard_value']
```

```
        # Example 4: Build a simple linear regression model
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

        model = LinearRegression()
        model.fit(X_train, y_train)

        y_pred = model.predict(X_test)

        # Example 5: Evaluate the model
        mse = mean_squared_error(y_test, y_pred)  # No squared=False needed
        r2 = r2_score(y_test, y_pred)

        print(f"Mean Squared Error: {mse}")
        print(f"R-squared: {r2}")
    else:
        print("Not enough valid data to build a model.")
else:
    print("No data to process.")
```

4. Xử lý lỗi (Error Handling):

a. ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard_value ~ '^[0-9\.]+$'

English:

This error occurs because you're trying to use a regular expression match (~) on a numeric column (`act.standard_value`). The database is expecting a string for the right-hand side of the ~ operator, but it's not getting it.

Solution: Remove the regular expression and rely on the `standard_value IS NOT NULL` condition. If you need to filter for numeric values, ensure the `standard_value` column is actually of a numeric type. You can cast it if necessary. However, a more robust approach is to validate and convert the data in your Python script.

Vietnamese:

Lỗi này xảy ra vì bạn đang cố gắng sử dụng so khớp biểu thức chính quy (~) trên một cột số (`act.standard_value`). Cơ sở dữ liệu mong đợi một chuỗi ở phía bên phải của toán tử ~, nhưng nó không nhận được.

Giải pháp: Xóa biểu thức chính quy và dựa vào điều kiện `standard_value IS NOT NULL`. Nếu bạn cần lọc các giá trị số, hãy đảm bảo cột `standard_value` thực sự là kiểu số. Bạn có thể ép kiểu nếu cần. Tuy nhiên, một cách tiếp cận mạnh mẽ hơn là xác thực và chuyển đổi dữ liệu trong tập lệnh Python của bạn.

Corrected SQL (Example 1):

```sql
SELECT  md.chembl_id,
        cs.canonical_smiles,
        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'
  AND   act.standard_units = 'nM'
  AND   act.standard_value IS NOT NULL  -- Remove the regex
LIMIT 100;
```

b. phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean_squared_error

English:

Older versions of scikit-learn do not support the `squared=False` parameter in the `mean_squared_error` function. This parameter is used to return the Root Mean Squared Error (RMSE) directly.

Solution: Remove the `squared=False` parameter. The function will return the Mean Squared Error (MSE). If you need the RMSE, calculate it manually by taking the square root of the MSE.

Vietnamese:

Các phiên bản cũ của scikit-learn không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`. Tham số này được sử dụng để trả về Root Mean Squared Error (RMSE) trực tiếp.

Giải pháp: Xóa tham số `squared=False`. Hàm sẽ trả về Mean Squared Error (MSE). Nếu bạn cần RMSE, hãy tính toán nó thủ công bằng cách lấy căn bậc hai của MSE.

Corrected Python (Example 5):

```
mse = mean_squared_error(y_test, y_pred)  # Remove squared=False
rmse = np.sqrt(mse)  # Calculate RMSE manually
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
```

5. Giới hạn dữ liệu (Data Limitation):

Tất cả các ví dụ trên đều giới hạn dữ liệu ở 100 dòng đầu tiên để đảm bảo hiệu suất. Bạn có thể thay đổi giá trị này nếu cần.

Lưu ý quan trọng:

- **Đường dẫn:** Hãy chắc chắn rằng bạn đã điều chỉnh đường dẫn `base_path` và `csv_file` cho phù hợp với cấu trúc thư mục của bạn.
- **Cột dữ liệu:** Kiểm tra xem tên cột trong file CSV của bạn có trùng khớp với tên cột được sử dụng trong code Python hay không (ví dụ: `canonical_smiles`, `standard_value`).
- **Loại hoạt tính:** Chọn loại hoạt tính phù hợp với mục tiêu nghiên cứu của bạn (ví dụ: IC50, Ki, EC50).
- **Tiền xử lý dữ liệu:** Dữ liệu ChEMBL có thể chứa nhiều giá trị khác nhau cho cùng một hợp chất. Hãy xem xét việc tiền xử lý dữ liệu để chuẩn hóa các giá trị này (ví dụ: lấy giá trị trung bình, xử lý các giá trị ngoại lệ).
- **Kiểm tra:** Luôn kiểm tra kết quả của bạn để đảm bảo tính chính xác và hợp lý.

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