PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Thuốc có nhiều dữ liệu hoạt tính nhất

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu Chembl 35 này. Với vai trò là chuyên gia nghiên cứu và phát triển dược học, tôi sẽ cung cấp hướng dẫn chi tiết, code mẫu SQL và Python, cùng với các ví dụ để bạn có thể triển khai dự án một cách hiệu quả.

Tổng quan dự án:

Dự án của bạn tập trung vào việc khai thác dữ liệu từ Chembl 35 kết hợp với RDKit để hỗ trợ nghiên cứu và phát triển thuốc. Bạn đã thiết lập cơ sở dữ liệu PostgreSQL (psql) và sử dụng Jupyter Notebook để phân tích dữ liệu.

Phân tích và Hướng dẫn:

1. **Mô hình Phân tích:**
    - **Mục tiêu:** Phân tích dữ liệu Chembl 35 để xác định các hợp chất có hoạt tính sinh học tiềm năng.
    - **Phương pháp:**
        - **Lọc và Chuẩn bị Dữ liệu:** Sử dụng SQL để truy vấn và lọc dữ liệu từ cơ sở dữ liệu Chembl 35, tập trung vào các thông tin quan trọng như cấu trúc hóa học (SMILES), giá trị hoạt tính (IC50, Ki, Kd), và mục tiêu tác động (targets).
        - **Tính Toán Đặc Trưng Phân Tử:** Sử dụng RDKit để tính toán các đặc trưng phân tử (molecular descriptors) từ cấu trúc SMILES, ví dụ: trọng lượng phân tử, logP, số lượng vòng, v.v.
        - **Phân Tích Thống Kê và Học Máy:**
            - **Phân tích tương quan:** Tìm hiểu mối quan hệ giữa các đặc trưng phân tử và giá trị hoạt tính.
            - **Mô hình hóa QSAR/QSPR:** Xây dựng mô hình dự đoán hoạt tính dựa trên các đặc trưng phân tử. Có thể sử dụng các thuật toán như Linear Regression, Random Forest, Support Vector Machines (SVM).
            - **Phân cụm (Clustering):** Nhóm các hợp chất có đặc trưng tương đồng để xác định các "scaffold" tiềm năng.
    - **Đánh giá mô hình:** Sử dụng các chỉ số như R-squared, RMSE, MAE để đánh giá hiệu suất của mô hình.

English Translation:

- **Goal:** Analyze Chembl 35 data to identify compounds with potential biological activity.
- **Methods:**
    - **Data Filtering and Preparation:** Use SQL to query and filter data from the Chembl 35 database, focusing on key information such as chemical structures (SMILES), activity values (IC50, Ki, Kd), and targets.
    - **Molecular Descriptor Calculation:** Use RDKit to calculate molecular descriptors from SMILES structures, e.g., molecular weight, logP, number of rings, etc.
    - **Statistical Analysis and Machine Learning:**
        - **Correlation analysis:** Explore the relationship between molecular descriptors and activity values.

- **QSAR/QSPR modeling:** Build models to predict activity based on molecular descriptors. Algorithms like Linear Regression, Random Forest, and Support Vector Machines (SVM) can be used.
- **Clustering:** Group compounds with similar features to identify potential scaffolds.

- **Model evaluation:** Use metrics like R-squared, RMSE, MAE to evaluate model performance.

2. **Hướng dẫn Song ngữ:**

   - **SQL:**
     - Sử dụng các câu lệnh `SELECT`, `FROM`, `WHERE` để truy vấn dữ liệu.
     - Sử dụng `LIKE` hoặc `~` để lọc dữ liệu dựa trên chuỗi (lưu ý lỗi bạn gặp phải).
     - Sử dụng `LIMIT` để giới hạn số lượng bản ghi trả về.
   - **Python:**
     - Sử dụng thư viện `psycopg2` để kết nối với cơ sở dữ liệu PostgreSQL.
     - Sử dụng `pandas` để đọc dữ liệu từ file CSV và xử lý dữ liệu.
     - Sử dụng `RDKit` để tính toán đặc trưng phân tử.
     - Sử dụng `scikit-learn` để xây dựng và đánh giá mô hình học máy.

   English Translation:

   - **SQL:**
     - Use `SELECT`, `FROM`, `WHERE` statements to query data.
     - Use `LIKE` or `~` to filter data based on strings (note the error you encountered).
     - Use `LIMIT` to limit the number of records returned.
   - **Python:**
     - Use the `psycopg2` library to connect to the PostgreSQL database.
     - Use `pandas` to read data from CSV files and process the data.
     - Use `RDKit` to calculate molecular descriptors.
     - Use `scikit-learn` to build and evaluate machine learning models.

3. **Code SQL và Python (100 dòng dữ liệu):**

   SQL (query_chembl.sql):

```sql
-- Select 100 compounds with IC50 values against a specific target (e.g., CHEMBL205)
SELECT DISTINCT ON (mol.molregno)
    mol.molregno,
    mol.smiles,
    act.standard_value,
    act.standard_units
FROM
    compound_structures mol
JOIN
    activities act ON mol.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.chembl_id = 'CHEMBL205'  -- Replace with your target of interest
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
LIMIT 100;
```

   Python (notebook/Topic_CheMBL_35_42_1_data_extraction.ipynb):

```python
import os
import psycopg2
import pandas as pd

# Database connection details
db_host = "192.168.206.136"
db_name = "chembl_35"
db_user = "rd"
db_pass = "rd"

# Base path for the project
base_path = os.getcwd()  # Assuming you're running this from the project's root
directory

# Output file path
output_file = os.path.join(base_path, 'data', 'chembl_ic50_data.csv')

# SQL query (replace with your actual query)
sql_query = """
SELECT DISTINCT ON (mol.molregno)
    mol.molregno,
    mol.smiles,
    act.standard_value,
    act.standard_units
FROM
    compound_structures mol
JOIN
    activities act ON mol.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.chembl_id = 'CHEMBL205'  -- Replace with your target of interest
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
LIMIT 100;
"""

# Establish connection to the database
try:
    conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass)
    print("Connected to the database successfully!")
except psycopg2.Error as e:
    print(f"Unable to connect to the database: {e}")
    exit()

# Execute the query and fetch the data into a Pandas DataFrame
try:
    df = pd.read_sql_query(sql_query, conn)
    print("Data fetched successfully!")
except Exception as e:
    print(f"Error fetching data: {e}")
    conn.close()
    exit()

# Close the database connection
conn.close()
```

```python
# Save the DataFrame to a CSV file
try:
    df.to_csv(output_file, index=False)
    print(f"Data saved to {output_file}")
except Exception as e:
    print(f"Error saving data to CSV: {e}")

print(df.head())
```

Giải thích:

- Đoạn code Python này kết nối đến cơ sở dữ liệu PostgreSQL, thực hiện câu truy vấn SQL và lưu kết quả vào file CSV.
- Bạn cần thay thế `'CHEMBL205'` bằng mã CHEMBL ID của mục tiêu bạn quan tâm.
- Đảm bảo thư viện `psycopg2` đã được cài đặt (`pip install psycopg2-binary`).

English Explanation:

- This Python code connects to the PostgreSQL database, executes the SQL query, and saves the results to a CSV file.
- You need to replace `'CHEMBL205'` with the CHEMBL ID of your target of interest.
- Ensure the `psycopg2` library is installed (`pip install psycopg2-binary`).

4. **Sửa lỗi:**

- **Lỗi `operator does not exist: numeric ~ unknown`:** Lỗi này xảy ra khi bạn sử dụng toán tử `~` (regex match) với kiểu dữ liệu số. Thay vì sử dụng regex, hãy chuyển đổi cột `act.standard_value` sang kiểu text và sử dụng `LIKE` để so sánh:

```
AND CAST(act.standard_value AS TEXT) LIKE '%.%'  -- Check if it contains a decimal point
AND CAST(act.standard_value AS TEXT) NOT LIKE '%[^0-9.]%' --Check if contains just number and '.'
```

- **Lỗi `squared=False` trong `mean_squared_error`:** Nâng cấp phiên bản `scikit-learn` lên phiên bản mới nhất (`pip install scikit-learn --upgrade`) hoặc loại bỏ tham số `squared=False` nếu bạn không cần tính RMSE (căn bậc hai của MSE).

5. Ví dụ Code SQL và Python Mẫu:

Ví dụ 1: Lọc các hợp chất có IC50 < 100 nM cho một mục tiêu cụ thể.

SQL:

```sql
SELECT
    mol.molregno,
    mol.smiles,
    act.standard_value
FROM
    compound_structures mol
JOIN
    activities act ON mol.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'
    AND act.standard_value < 100
LIMIT 10;
```

Python:

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

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

df = pd.read_csv('data/chembl_ic50_data.csv')
df['logP'] = df['smiles'].apply(calculate_logp)
print(df.head())
```

Ví dụ 2: Tính trọng lượng phân tử cho các hợp chất.

SQL: Không cần thiết, có thể thực hiện trực tiếp trong Python.

Python:

```python
from rdkit import Chem
from rdkit.Chem import Descriptors

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

df = pd.read_csv('data/chembl_ic50_data.csv')
df['mol_weight'] = df['smiles'].apply(calculate_mw)
print(df.head())
```

Ví dụ 3: Tìm các hợp chất có chứa một motif cụ thể.

SQL:

```sql
SELECT
    mol.molregno,
    mol.smiles
FROM
    compound_structures mol
WHERE
    mol.smiles LIKE '%[Cl]c1ccccc1%';  -- Tìm các hợp chất có vòng benzene chứa clo
```

Python:

```python
import pandas as pd

df = pd.read_csv('data/chembl_ic50_data.csv')
df_filtered = df[df['smiles'].str.contains(r'\[Cl]c1ccccc1')]
print(df_filtered.head())
```

Ví dụ 4: Phân tích cơ bản về giá trị IC50.

SQL: Không cần thiết, có thể thực hiện trực tiếp trong Python.

Python:

```python
import pandas as pd
```

```
df = pd.read_csv('data/chembl_ic50_data.csv')
print(df['standard_value'].describe())
```

Ví dụ 5: Chuẩn hóa dữ liệu hoạt tính (ví dụ pIC50).

SQL: Không cần thiết, có thể thực hiện trực tiếp trong Python.

Python:

```
import pandas as pd
import numpy as np

df = pd.read_csv('data/chembl_ic50_data.csv')
df = df.dropna(subset=['standard_value'])
df['pIC50'] = -np.log10(df['standard_value'] / 1e9)  # Convert IC50 (nM) to pIC50
print(df.head())
```

English Translation of Examples:

Example 1: Filter compounds with IC50 < 100 nM for a specific target.

SQL:

```
SELECT
    mol.molregno,
    mol.smiles,
    act.standard_value
FROM
    compound_structures mol
JOIN
    activities act ON mol.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'
    AND act.standard_value < 100
LIMIT 10;
```

Python:

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

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

df = pd.read_csv('data/chembl_ic50_data.csv')
df['logP'] = df['smiles'].apply(calculate_logp)
print(df.head())
```

Example 2: Calculate the molecular weight for compounds.

SQL: Not needed, can be done directly in Python.

Python:

```python
from rdkit import Chem
from rdkit.Chem import Descriptors

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

df = pd.read_csv('data/chembl_ic50_data.csv')
df['mol_weight'] = df['smiles'].apply(calculate_mw)
print(df.head())
```

Example 3: Find compounds containing a specific motif.

SQL:

```sql
SELECT
    mol.molregno,
    mol.smiles
FROM
    compound_structures mol
WHERE
    mol.smiles LIKE '%[Cl]c1ccccc1%';  -- Find compounds with a benzene ring containing chlorine
```

Python:

```python
import pandas as pd

df = pd.read_csv('data/chembl_ic50_data.csv')
df_filtered = df[df['smiles'].str.contains(r'\[Cl]c1ccccc1')]
print(df_filtered.head())
```

Example 4: Basic analysis of IC50 values.

SQL: Not needed, can be done directly in Python.

Python:

```python
import pandas as pd

df = pd.read_csv('data/chembl_ic50_data.csv')
print(df['standard_value'].describe())
```

Example 5: Normalize activity data (e.g., pIC50).

SQL: Not needed, can be done directly in Python.

Python:

```python
import pandas as pd
import numpy as np

df = pd.read_csv('data/chembl_ic50_data.csv')
df = df.dropna(subset=['standard_value'])
df['pIC50'] = -np.log10(df['standard_value'] / 1e9)  # Convert IC50 (nM) to pIC50
print(df.head())
```

Lưu ý quan trọng:

- Hãy điều chỉnh các câu truy vấn SQL và đoạn code Python cho phù hợp với mục tiêu nghiên cứu cụ thể của bạn.
- Kiểm tra kỹ dữ liệu sau khi trích xuất và làm sạch để đảm bảo tính chính xác.
- Sử dụng các công cụ trực quan hóa dữ liệu (ví dụ: Matplotlib, Seaborn) để khám phá dữ liệu và trình bày kết quả.

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