PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Thiết kế thuốc dựa trên ligand từ ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit. Với vai trò là một chuyên gia R&D dược học, tôi hiểu tầm quan trọng của việc khai thác dữ liệu hóa sinh để đẩy nhanh quá trình phát triển thuốc.

Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu SQL và Python, cùng với các lưu ý quan trọng để bạn có thể triển khai dự án một cách hiệu quả.

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

Mục tiêu của bạn là phân tích dữ liệu ChEMBL 35, một cơ sở dữ liệu lớn chứa thông tin về các phân tử hóa học và hoạt tính sinh học của chúng. Kết hợp với RDKit, một bộ công cụ hóa tin học mạnh mẽ, bạn có thể thực hiện nhiều phân tích khác nhau, bao gồm:

- **Phân tích Hoạt tính (Activity Analysis):** Xác định các phân tử có hoạt tính cao đối với một mục tiêu sinh học cụ thể. Điều này bao gồm việc lọc dữ liệu, chuẩn hóa giá trị hoạt tính, và xác định các hợp chất "hit".
- **Phân tích SAR (Structure-Activity Relationship):** Nghiên cứu mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học. Điều này có thể giúp bạn xác định các nhóm chức quan trọng cho hoạt tính và thiết kế các phân tử mới có hoạt tính tốt hơn.
- **Phân tích QSAR/QSPR (Quantitative Structure-Activity/Property Relationship):** Xây dựng các mô hình toán học để dự đoán hoạt tính hoặc tính chất của một phân tử dựa trên cấu trúc của nó. Điều này có thể giúp bạn sàng lọc các phân tử tiềm năng một cách nhanh chóng và hiệu quả.
- **Phân tích Tương đồng (Similarity Analysis):** Tìm kiếm các phân tử tương tự về cấu trúc hoặc tính chất với một phân tử đã biết. Điều này có thể giúp bạn tìm ra các phân tử "me-too" hoặc "me-better".
- **Phân tích Đa dạng (Diversity Analysis):** Đánh giá sự đa dạng của một bộ sưu tập các phân tử. Điều này có thể giúp bạn chọn ra các phân tử đại diện để thử nghiệm hoặc mua.

2. Hướng dẫn Song ngữ (Bilingual Guide)

2.1 Chuẩn bị Dữ liệu (Data Preparation)

- **SQL:** Sử dụng SQL để truy vấn và lọc dữ liệu từ cơ sở dữ liệu ChEMBL 35.
- **Python:** Sử dụng Python và thư viện pandas để đọc dữ liệu từ file CSV, tiền xử lý dữ liệu, và chuẩn bị cho các phân tích tiếp theo.

2.2 Tính toán Descriptor (Descriptor Calculation)

- **RDKit:** Sử dụng RDKit để tính toán các descriptor hóa học (ví dụ: trọng lượng phân tử, số lượng liên kết hydro, logP) từ cấu trúc của các phân tử.

2.3 Phân tích và Mô hình hóa (Analysis and Modeling)

- **Python:** Sử dụng các thư viện như scikit-learn để xây dựng các mô hình QSAR/QSPR, thực hiện phân tích tương đồng, hoặc phân tích đa dạng.

3. Code Mẫu (Code Examples)

3.1 SQL

```sql
-- Lấy 100 dòng dữ liệu hoạt tính của một mục tiêu cụ thể (ví dụ: CHEMBL205)
-- Get 100 rows of activity data for a specific target (e.g., CHEMBL205)
SELECT
    act.molregno,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    md.chembl_id,
    md.pref_name
FROM activities act
JOIN target_dictionary td ON act.tid = td.tid
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE td.chembl_id = 'CHEMBL205' -- Thay đổi CHEMBL ID tùy theo mục tiêu của bạn
  AND act.standard_type = 'IC50'
  AND act.standard_relation = '='
  AND act.standard_value IS NOT NULL
  AND act.standard_units = 'nM'
LIMIT 100;
```

Lưu ý:

- Sửa lỗi `ERROR: operator does not exist: numeric ~ unknown`: Lỗi này xảy ra do bạn đang cố gắng so sánh một cột kiểu số (`act.standard_value`) với một chuỗi (`'^[0-9\.]+$'`). Trong PostgreSQL, bạn không cần sử dụng toán tử `~` để kiểm tra xem một giá trị có phải là số hay không. Bạn có thể bỏ qua điều kiện này hoặc sử dụng hàm `regexp_match` nếu bạn thực sự muốn kiểm tra định dạng của chuỗi. Tuy nhiên, trong trường hợp này, có vẻ như điều kiện này không cần thiết.

3.2 Python

```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

# Đường dẫn cơ sở của dự án (Base path of the project)
base_path = "."  # Thay đổi nếu cần (Change if needed)
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Đọc dữ liệu từ file CSV (Read data from CSV file)
data = pd.read_csv(os.path.join(data_path, "chembl_205_ic50_100.csv"))

# Hiển thị một vài dòng đầu tiên của dữ liệu (Display the first few rows of data)
print(data.head())

# Tạo cột SMILES (Create SMILES column - assuming you have a 'molregno' column that
can be linked to the 'compound_structures' table in ChEMBL)
# **This is a placeholder.  You'll need to adapt this based on how you get SMILES from
your ChEMBL data.**
# For example, if you have a 'smiles' column directly:
# data['SMILES'] = data['smiles']
# Or, if you need to fetch it from a database:
# def get_smiles(molregno):
#   # Connect to your database and fetch the SMILES based on molregno
#   # This is just example code, adapt to your database connection and query
#   import psycopg2
```

```
#   conn = psycopg2.connect(host="192.168.206.136", database="chembl_35", user="rd",
password="rd")
#   cur = conn.cursor()
#   cur.execute("SELECT canonical_smiles FROM compound_structures WHERE molregno =
%s", (molregno,))
#   smiles = cur.fetchone()[0]  # Assuming canonical_smiles is the correct column
#   conn.close()
#   return smiles
#
# data['SMILES'] = data['molregno'].apply(get_smiles)

# For simplicity, let's assume you have a CSV with SMILES already
data = data.dropna(subset=['SMILES']) #Drop if there are NaN values

# Tính toán descriptor (Calculate descriptors)
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    descriptors['MW'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    descriptors['HBD'] = Descriptors.NumHDonors(mol)
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
    return pd.Series(descriptors)

data[['MW', 'LogP', 'HBD', 'HBA']] = data['SMILES'].apply(calculate_descriptors)

# Chuẩn bị dữ liệu cho mô hình (Prepare data for the model)
data = data.dropna() # Remove rows with missing descriptor values
X = data[['MW', 'LogP', 'HBD', 'HBA']]
y = data['standard_value']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra (Split data into training and
testing sets)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Xây dựng mô hình hồi quy tuyến tính (Build a linear regression model)
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra (Make predictions 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:

- Sửa lỗi `ValueError: Unknown label type: 'continuous'`: Lỗi này thường xảy ra khi bạn sử dụng các thuật toán phân loại (classification) với dữ liệu mục tiêu (target variable) là liên tục (continuous). Trong trường hợp này, bạn đang dự đoán giá trị IC50, là một giá trị liên tục, vì vậy bạn nên sử dụng các thuật toán hồi quy (regression) thay vì phân loại. Tôi đã sửa code để sử dụng `LinearRegression`, một thuật toán hồi quy tuyến tính.

- Sửa lỗi `AttributeError: 'collections.OrderedDict' object has no attribute 'viewkeys'`: Lỗi này liên quan đến phiên bản scikit-learn. Bạn có thể thử nâng cấp scikit-learn lên phiên bản mới nhất hoặc sử dụng các phiên bản scikit-learn cũ hơn.
- Sửa lỗi `TypeError: mean_squared_error() got an unexpected keyword argument 'squared'`: Lỗi này xảy ra do phiên bản scikit-learn cũ không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`. Nếu bạn đang sử dụng một phiên bản scikit-learn cũ, hãy bỏ tham số `squared=False` đi. Phiên bản mới mặc định trả về MSE (squared=True). Nếu bạn muốn RMSE, hãy tính căn bậc hai của MSE.

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

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

```sql
-- SQL
SELECT
    act.molregno,
    md.chembl_id,
    act.standard_value
FROM activities act
JOIN target_dictionary td ON act.tid = td.tid
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE td.chembl_id = 'CHEMBL205'
  AND act.standard_type = 'IC50'
  AND act.standard_relation = '='
  AND act.standard_value IS NOT NULL
  AND act.standard_units = 'nM'
  AND act.standard_value < 100
LIMIT 100;
```

```python
# Python
import pandas as pd

# Assuming you have a CSV file named 'chembl_ic50.csv' with 'standard_value' column
df = pd.read_csv("chembl_ic50.csv")
df_filtered = df[df['standard_value'] < 100]
print(df_filtered.head())
```

Ví dụ 2: Tính toán LogP sử dụng RDKit

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

smiles = 'CC(=O)Oc1ccccc1C(=O)O'  # Aspirin
mol = Chem.MolFromSmiles(smiles)
logp = Descriptors.MolLogP(mol)
print(f"LogP for Aspirin: {logp}")
```

Ví dụ 3: Phân tích tần suất của các khung phân tử (Molecular Frameworks)

```python
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem.Scaffolds import MurckoScaffold
import pandas as pd

# Assuming you have a list of SMILES strings
smiles_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'C1CCCCC1']
frameworks = []
for smiles in smiles_list:
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        core = MurckoScaffold.GetScaffoldForMol(mol)
```

```
        frameworks.append(Chem.MolToSmiles(core))

framework_counts = pd.Series(frameworks).value_counts()
print(framework_counts)
```

Ví dụ 4: Sử dụng Morgan Fingerprint cho phân tích tương đồng

```
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit import DataStructs

smiles1 = 'CC(=O)Oc1ccccc1C(=O)O'  # Aspirin
smiles2 = 'CC(=O)Oc1ccccc1C(=O)C'  # Modified Aspirin
mol1 = Chem.MolFromSmiles(smiles1)
mol2 = Chem.MolFromSmiles(smiles2)

fp1 = AllChem.GetMorganFingerprintAsBitVect(mol1, 2, nBits=1024)
fp2 = AllChem.GetMorganFingerprintAsBitVect(mol2, 2, nBits=1024)

similarity = DataStructs.TanimotoSimilarity(fp1, fp2)
print(f"Tanimoto Similarity: {similarity}")
```

Ví dụ 5: Kết nối đến cơ sở dữ liệu ChEMBL sử dụng Python

```
import psycopg2

try:
    conn = psycopg2.connect(host="192.168.206.136", database="chembl_35", user="rd",
password="rd")
    cur = conn.cursor()

    # Execute a query
    cur.execute("SELECT COUNT(*) FROM molecule_dictionary;")
    count = cur.fetchone()[0]
    print(f"Number of molecules in ChEMBL: {count}")

    conn.close()

except psycopg2.Error as e:
    print(f"Error connecting to the database: {e}")
```

5. Lưu ý Quan trọng (Important Notes)

- **Phiên bản thư viện:** Đảm bảo bạn sử dụng các phiên bản thư viện tương thích.
- **Xử lý lỗi:** Xử lý các lỗi có thể xảy ra trong quá trình đọc dữ liệu, tính toán descriptor, và xây dựng mô hình.
- **Kiểm tra dữ liệu:** Luôn kiểm tra dữ liệu của bạn để đảm bảo tính chính xác và đầy đủ.
- **Tài liệu RDKit:** Tham khảo tài liệu RDKit để biết thêm thông tin về các descriptor và thuật toán có sẵn.

Tôi hy vọng những hướng dẫn này sẽ giúp bạn triển khai dự án của mình một cách thành công. Nếu bạn có bất kỳ câu hỏi nào, đừng ngần ngại hỏi!