PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Kết nối ChEMBL với DrugBank để tái định hướng thuốc

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 sẽ cung cấp cho bạn hướng dẫn chi tiết, mã nguồn SQL và Python mẫu, cùng với những lưu ý quan trọng để đảm bảo dự án của bạn diễn ra suôn sẻ.

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

Dựa trên thông tin bạn cung cấp, chúng ta sẽ xây dựng một quy trình phân tích dữ liệu tuân theo các bước sau:

1. **Data Extraction (Trích xuất dữ liệu):** Sử dụng SQL để truy vấn cơ sở dữ liệu ChEMBL 35 trên PostgreSQL và lưu kết quả vào các file CSV.
2. **Data Preprocessing (Tiền xử lý dữ liệu):** Sử dụng Python (với thư viện RDKit và Pandas) để làm sạch, chuyển đổi và chuẩn hóa dữ liệu. Các bước có thể bao gồm:
    - Xử lý các giá trị missing.
    - Chuyển đổi các giá trị hoạt tính (IC50, Ki, EC50) sang pChEMBL.
    - Tính toán các đặc tính lý hóa (physicochemical properties) và fingerprint phân tử (molecular fingerprints) bằng RDKit.
3. **Exploratory Data Analysis (EDA) (Phân tích khám phá dữ liệu):** Sử dụng Python để khám phá dữ liệu, tìm kiếm các xu hướng và mối quan hệ giữa các biến. Các kỹ thuật có thể bao gồm:
    - Vẽ biểu đồ phân phối của các biến.
    - Tính toán các thống kê mô tả.
    - Phân tích tương quan.
4. **Model Building (Xây dựng mô hình):** Sử dụng các thuật toán học máy (ví dụ: Random Forest, Support Vector Machines) để xây dựng các mô hình dự đoán hoạt tính của các hợp chất.
5. **Model Evaluation (Đánh giá mô hình):** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp (ví dụ: RMSE, R-squared, AUC).

2. Hướng dẫn song ngữ (Bilingual Instructions)

2.1. Kết nối cơ sở dữ liệu (Connecting to the Database)

Tiếng Việt:

Để kết nối đến cơ sở dữ liệu ChEMBL 35 của bạn, bạn cần sử dụng thư viện `psycopg2` trong Python. Đảm bảo rằng bạn đã cài đặt thư viện này bằng lệnh `pip install psycopg2`.

English:

To connect to your ChEMBL 35 database, you need to use the `psycopg2` library in Python. Make sure you have installed this library using the command `pip install psycopg2`.

```
import psycopg2

# Database credentials
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35'
```

```
}

def connect_to_db(params):
    """Connects to the PostgreSQL database."""
    try:
        conn = psycopg2.connect(**params)
        return conn
    except psycopg2.Error as e:
        print(f"Error connecting to the database: {e}")
        return None

# Establish connection
connection = connect_to_db(db_params)

if connection:
    print("Successfully connected to the database!")
    # Perform database operations here
    connection.close() # remember to close connection after done
else:
    print("Failed to connect to the database.")
```

2.2. Trích xuất dữ liệu với SQL (Data Extraction with SQL)

Tiếng Việt:

Sử dụng câu lệnh SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35. Lưu ý, để tránh lỗi "operator does not exist: numeric ~ unknown", hãy sử dụng hàm `CAST` để chuyển đổi kiểu dữ liệu của cột `standard_value` sang kiểu `TEXT` trước khi so sánh với regular expression.

English:

Use SQL queries to extract data from the ChEMBL 35 database. Note, to avoid the error "operator does not exist: numeric ~ unknown", use the `CAST` function to convert the data type of the `standard_value` column to `TEXT` before comparing with the regular expression.

```
-- Lấy 100 dòng dữ liệu
-- Get 100 rows of data
SELECT DISTINCT
    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'
  AND act.standard_relation = '='
  AND act.standard_units = 'nM'
  AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$' -- fix error here
LIMIT 100;
```

2.3. Tiền xử lý dữ liệu với Python (Data Preprocessing with Python)

Tiếng Việt:

Sử dụng thư viện Pandas để đọc dữ liệu từ file CSV, RDKit để tính toán fingerprint phân tử và pChEMBL, và Scikit-learn cho các bước tiền xử lý khác.

English:

Use the Pandas library to read data from CSV files, RDKit to calculate molecular fingerprints and pChEMBL, and Scikit-learn for other preprocessing steps.

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
import numpy as np

def calculate_pchembl(standard_value):
    """Calculates pChEMBL value."""
    return -np.log10(standard_value * 1e-9)

def calculate_fingerprint(smiles):
    """Calculates Morgan fingerprint using RDKit."""
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
    else:
        return None

# Example usage
csv_file_path = '../data/your_file.csv'  # Replace with your actual file path
df = pd.read_csv(csv_file_path)

# Calculate pChEMBL (assuming IC50 in nM)
df['pchembl_value'] = df['standard_value'].apply(calculate_pchembl)

# Calculate molecular fingerprints
df['mol'] = df['canonical_smiles'].apply(Chem.MolFromSmiles)
df = df[df['mol'].notna()]
df['fingerprint'] = df['canonical_smiles'].apply(calculate_fingerprint)
df = df[df['fingerprint'].notna()]

print(df.head())
```

2.4 Xử lý lỗi phiên bản Scikit-learn (Handling Scikit-learn Version Error)

Tiếng Việt:

Nếu bạn gặp lỗi liên quan đến tham số `squared=False` trong hàm `mean_squared_error`, hãy kiểm tra phiên bản Scikit-learn của bạn. Nếu phiên bản cũ, bạn có thể nâng cấp lên phiên bản mới hơn hoặc bỏ tham số `squared=False` (khi đó, kết quả sẽ là MSE thay vì RMSE).

English:

If you encounter an error related to the `squared=False` parameter in the `mean_squared_error` function, check your Scikit-learn version. If the version is old, you can upgrade to a newer version or remove the `squared=False` parameter (in which case the result will be MSE instead of RMSE).

```
from sklearn.metrics import mean_squared_error

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

# Try to calculate RMSE; if it fails, calculate MSE instead
try:
    rmse = mean_squared_error(y_true, y_predicted, squared=False)
    print(f"RMSE: {rmse}")
```

```
except TypeError:
    mse = mean_squared_error(y_true, y_predicted)
    print(f"MSE: {mse}. Consider upgrading scikit-learn for RMSE support.")
```

3. Ví dụ mã SQL và Python mẫu (Example SQL and Python Code Samples)

3.1. SQL Examples

```
-- 1. Lấy danh sách các hợp chất có hoạt tính IC50 dưới 100 nM
-- 1. Get a list of compounds with IC50 activity below 100 nM
SELECT md.chembl_id, cs.canonical_smiles
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_value < 100 AND act.standard_units =
'nM';

-- 2. Lấy số lượng các hợp chất cho mỗi giá trị standard_type
-- 2. Get the number of compounds for each standard_type
SELECT act.standard_type, COUNT(DISTINCT md.chembl_id) AS num_compounds
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
GROUP BY act.standard_type;

-- 3. Lấy các hợp chất có khối lượng phân tử nằm trong khoảng 400-500 Da
-- 3. Get compounds with molecular weight between 400-500 Da
SELECT md.chembl_id, cs.canonical_smiles
FROM molecule_dictionary md
JOIN compound_structures cs ON md.molregno = cs.molregno
WHERE md.mw_freebase BETWEEN 400 AND 500;

-- 4. Lấy danh sách các mục tiêu (targets) liên quan đến một hợp chất cụ thể
-- 4. Get a list of targets related to a specific compound
SELECT td.chembl_id, td.pref_name
FROM target_dictionary td
JOIN assays a ON td.tid = a.tid
JOIN activities act ON a.assay_id = act.assay_id
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE md.chembl_id = 'CHEMBL188'; -- Thay thế bằng Chembl ID bạn muốn

-- 5. Lấy danh sách các protein có hoạt tính Ki
-- 5. Get the list of proteins with Ki activities
SELECT td.chembl_id, td.pref_name
FROM target_dictionary td
JOIN assays a ON td.tid = a.tid
JOIN activities act ON a.assay_id = act.assay_id
WHERE act.standard_type = 'Ki';
```

3.2. Python Examples

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

# 1. Tính toán LogP cho một danh sách các SMILES
# 1. Calculate LogP for a list of SMILES
smiles_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']
logp_values = []

for smiles in smiles_list:
    mol = Chem.MolFromSmiles(smiles)
```

```
    if mol:
        logp = Descriptors.MolLogP(mol)
        logp_values.append(logp)
    else:
        logp_values.append(None)

print(f"LogP values: {logp_values}")

# 2. Đọc dữ liệu từ CSV và tính toán số vòng thơm
# 2. Read data from CSV and calculate the number of aromatic rings
csv_file = '../data/your_file.csv'
df = pd.read_csv(csv_file)

def calculate_aromatic_rings(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Chem.Lipinski.NumAromaticRings(mol)
    else:
        return None

df['aromatic_rings'] = df['canonical_smiles'].apply(calculate_aromatic_rings)
print(df[['canonical_smiles', 'aromatic_rings']].head())

# 3. Lọc các hợp chất tuân theo quy tắc 5 của Lipinski
# 3. Filter compounds that follow Lipinski's rule of 5
def lipinski_rule_of_five(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        mw = Descriptors.MolWt(mol)
        logp = Descriptors.MolLogP(mol)
        hbd = Chem.Lipinski.NumHDonors(mol)
        hba = Chem.Lipinski.NumHAcceptors(mol)
        return mw <= 500 and logp <= 5 and hbd <= 5 and hba <= 10
    else:
        return False

df['lipinski_compliant'] = df['canonical_smiles'].apply(lipinski_rule_of_five)
lipinski_compliant_df = df[df['lipinski_compliant']]
print(f"Number of Lipinski-compliant compounds: {len(lipinski_compliant_df)}")

# 4. Lưu các fingerprints vào một file
# 4. Save the fingerprints to a file
from rdkit.Chem import MACCSkeys
import pickle

def calculate_maccs_keys(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return MACCSkeys.GenMACCSKeys(mol)
    else:
        return None

df['maccs_keys'] = df['canonical_smiles'].apply(calculate_maccs_keys)
maccs_keys_list = df['maccs_keys'].tolist()

with open('../data/maccs_keys.pkl', 'wb') as f:
    pickle.dump(maccs_keys_list, f)

# 5. Tìm các cấu trúc con (substructure search)
```

```
# 5. Find substructures
from rdkit.Chem import Draw

substructure_smiles = 'c1ccccc1' # Benzen
substructure = Chem.MolFromSmiles(substructure_smiles)

def has_substructure(smiles, substructure):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return mol.HasSubstructMatch(substructure)
    else:
        return False

df['has_benzene'] = df['canonical_smiles'].apply(lambda x: has_substructure(x,
substructure))
benzene_containing_df = df[df['has_benzene']]

print(f"Number of compounds containing benzene: {len(benzene_containing_df)}")
```

Lưu ý quan trọng (Important Notes):

- **Đường dẫn file (File Paths):** Đảm bảo rằng các đường dẫn đến file CSV là chính xác. Sử dụng `os.path.join(base_path, ...)` để tạo đường dẫn một cách linh hoạt.
- **Phiên bản thư viện (Library Versions):** Kiểm tra và cập nhật các thư viện (RDKit, Pandas, Scikit-learn) để đảm bảo tính tương thích và tận dụng các tính năng mới nhất.
- **Xử lý lỗi (Error Handling):** Thêm các khối `try...except` để xử lý các lỗi có thể xảy ra trong quá trình xử lý dữ liệu.
- **Bộ nhớ (Memory):** Với bộ dữ liệu lớn, hãy cân nhắc sử dụng các kỹ thuật như chunking (đọc dữ liệu theo từng phần) để giảm thiểu việc sử dụng bộ nhớ.

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