PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Dòng AutoML cho dữ liệu ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. 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 thực hiện dự án một cách hiệu quả.

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

1.1. Mục tiêu (Objective)

- Phân tích dữ liệu ChEMBL 35 để khám phá các mối quan hệ giữa cấu trúc hóa học của các hợp chất và hoạt tính sinh học của chúng.
- Sử dụng RDKit để tạo các đặc trưng hóa học (chemical features) từ SMILES strings.
- Xây dựng mô hình học máy để dự đoán hoạt tính của các hợp chất mới dựa trên cấu trúc của chúng.

1.2. Dữ liệu (Data)

- Dữ liệu hoạt tính sinh học (bioactivity data) từ cơ sở dữ liệu ChEMBL 35.
- Cấu trúc hóa học của các hợp chất (SMILES strings).

1.3. Các bước thực hiện (Steps)

1. **Trích xuất dữ liệu (Data Extraction):** Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35 và lưu vào file CSV.
2. **Tiền xử lý dữ liệu (Data Preprocessing):**
    - Làm sạch và chuẩn hóa dữ liệu.
    - Loại bỏ các giá trị ngoại lệ (outliers) và các giá trị thiếu (missing values).
3. **Tạo đặc trưng hóa học (Feature Engineering):**
    - Sử dụng RDKit để chuyển đổi SMILES strings thành các đặc trưng hóa học như:
        - Molecular descriptors (ví dụ: trọng lượng phân tử, logP).
        - Fingerprints (ví dụ: Morgan fingerprints).
4. **Phân tích dữ liệu thăm dò (Exploratory Data Analysis - EDA):**
    - Thực hiện các phân tích thống kê mô tả.
    - Trực quan hóa dữ liệu để khám phá các xu hướng và mối quan hệ.
5. **Xây dựng mô hình học máy (Machine Learning Model Building):**
    - Lựa chọn mô hình phù hợp (ví dụ: Random Forest, Support Vector Machine).
    - Huấn luyện mô hình trên dữ liệu huấn luyện.
    - Đánh giá hiệu suất mô hình trên dữ liệu kiểm tra.
6. **Giải thích mô hình (Model Interpretation):**
    - Xác định các đặc trưng quan trọng nhất ảnh hưởng đến hoạt tính.
    - Rút ra các hiểu biết có giá trị để hỗ trợ quá trình thiết kế thuốc.

2. Hướng dẫn Song ngữ (Bilingual Guide)

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

- **Tiếng Việt:** Sử dụng thư viện `psycopg2` trong Python để kết nối đến cơ sở dữ liệu PostgreSQL.
- **English:** Use the `psycopg2` library in Python to connect to the PostgreSQL database.

```python
import psycopg2

# Thông tin kết nối (Connection details)
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35'
}

# Hàm kết nối (Connection function)
def connect_to_db(params):
    try:
        conn = psycopg2.connect(**params)
        print("Kết nối thành công đến cơ sở dữ liệu!")
        print("Successfully connected to the database!")
        return conn
    except psycopg2.Error as e:
        print(f"Lỗi kết nối: {e}")
        print(f"Connection error: {e}")
        return None

# Kết nối (Connect)
conn = connect_to_db(db_params)

# Đóng kết nối khi hoàn thành (Close connection when done)
if conn:
    conn.close()
    print("Đã đóng kết nối!")
    print("Connection closed!")
```

2.2. Truy vấn dữ liệu (Querying Data)

- **Tiếng Việt:** Sử dụng câu lệnh SQL để truy vấn dữ liệu từ các bảng trong cơ sở dữ liệu ChEMBL.
- **English:** Use SQL queries to retrieve data from tables in the ChEMBL database.

```sql
-- Ví dụ truy vấn (Example query)
SELECT
    md.molregno,
    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_units = 'nM'
LIMIT 100;
```

2.3. Tạo đặc trưng hóa học (Generating Chemical Features)

- **Tiếng Việt:** Sử dụng RDKit để tạo các đặc trưng hóa học từ SMILES strings.
- **English:** Use RDKit to generate chemical features from SMILES strings.

```python
from rdkit import Chem
from rdkit.Chem import AllChem
import numpy as np

def generate_morgan_fingerprint(smiles, radius=2, nBits=2048):
    """
    Generates a Morgan fingerprint for a given SMILES string.
```

```
    Args:
        smiles (str): SMILES string of the molecule.
        radius (int): Radius of the fingerprint.
        nBits (int): Number of bits in the fingerprint.

    Returns:
        numpy.array: Morgan fingerprint as a NumPy array.
    """
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    fingerprint = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits)
    return np.array(fingerprint)

# Ví dụ sử dụng (Example usage)
smiles = 'CC(=O)Oc1ccccc1C(=O)O'
fingerprint = generate_morgan_fingerprint(smiles)
print(fingerprint)
```

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

3.1. SQL (Trích xuất dữ liệu và lưu vào CSV - Extract data and save to CSV)

```
-- Lưu ý: Chạy trên pgAdmin và lưu kết quả vào file CSV (Note: Run on pgAdmin and save
the results to a CSV file)
-- Lưu file vào (Save file to): ../data/chembl_data.csv

SELECT
    md.molregno,
    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_units = 'nM'
  AND act.standard_value IS NOT NULL  -- Added check for NULL values
LIMIT 100;
```

3.2. Python (Đọc dữ liệu CSV, tạo fingerprints - Read CSV data, generate fingerprints)

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

# Đường dẫn cơ sở (Base path)
base_path = '..'  # Điều chỉnh nếu cần (Adjust if needed)
data_path = os.path.join(base_path, 'data', 'chembl_data.csv')

# Hàm tạo Morgan fingerprint (Function to generate Morgan fingerprint)
def generate_morgan_fingerprint(smiles, radius=2, nBits=2048):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    fingerprint = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits)
    return np.array(fingerprint)
```

```python
# Đọc dữ liệu (Read data)
try:
    df = pd.read_csv(data_path)
except FileNotFoundError:
    print(f"Không tìm thấy file: {data_path}")
    exit()

# Tiền xử lý dữ liệu (Data preprocessing)
df = df.dropna(subset=['canonical_smiles', 'standard_value'])

# Tạo fingerprints (Generate fingerprints)
df['fingerprint'] = df['canonical_smiles'].apply(generate_morgan_fingerprint)
df = df.dropna(subset=['fingerprint'])

print(df.head())
```

4. Ví dụ Code SQL và Python mẫu (Example SQL and Python Code Snippets)

4.1. SQL - Truy vấn hoạt tính theo khoảng giá trị (Query activities within a range)

```sql
SELECT
    md.molregno,
    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 BETWEEN 100 AND 1000
LIMIT 100;
```

4.2. Python - Tính toán các descriptor (Calculate Descriptors)

```python
from rdkit.Chem import Descriptors

def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {
        'MW': Descriptors.MolWt(mol),
        'logP': Descriptors.MolLogP(mol),
        'HBA': Descriptors.NumHAcceptors(mol),
        'HBD': Descriptors.NumHDonors(mol)
    }
    return descriptors

# Ví dụ sử dụng (Example usage)
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)
print(df['descriptors'].head())
```

4.3. SQL - Truy vấn theo mục tiêu (Target Specific Activity)

```sql
SELECT
    md.molregno,
    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.pref_name = 'Acetylcholinesterase'
  AND act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
LIMIT 100;
```

4.4. Python - Mô hình hóa (Modeling)

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Chuẩn bị dữ liệu (Prepare data)
X = np.stack(df['fingerprint'].values)
y = df['standard_value'].values

# Chia dữ liệu (Split data)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Huấn luyện mô hình (Train model)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Dự đoán (Predict)
y_pred = model.predict(X_test)

# Đánh giá (Evaluate)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'MSE: {mse}')
print(f'R2: {r2}')
```

4.5. SQL - Tìm kiếm cấu trúc con (Substructure Search)

```
SELECT
    md.molregno,
    cs.canonical_smiles
FROM molecule_dictionary md
JOIN compound_structures cs ON md.molregno = cs.molregno
WHERE cs.canonical_smiles LIKE '%C(=O)N%'  -- Tìm kiếm các hợp chất chứa amide bond
LIMIT 100;
```

5. Lưu ý về lỗi (Error Handling)

a. ERROR: operator does not exist: numeric ~ unknown

- **Nguyên nhân:** Lỗi này xảy ra khi bạn cố gắng sử dụng toán tử ~ (regex match) trên một cột số (`standard_value`). PostgreSQL không tự động chuyển đổi kiểu dữ liệu trong trường hợp này.
- **Giải pháp:** Chuyển đổi cột số sang kiểu text trước khi sử dụng ~, hoặc sử dụng các hàm so sánh số học trực tiếp.
- **Solution:** Cast the numeric column to text before using ~, or use direct numerical comparison functions.

```
-- Sửa lỗi (Corrected SQL)
SELECT
    md.molregno,
    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 CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'  -- Chuyển đổi sang text (Cast to text)
LIMIT 100;
```

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

- **Nguyên nhân:** Phiên bản scikit-learn bạn đang sử dụng đã cũ và không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`.
- **Giải pháp:** Nâng cấp scikit-learn lên phiên bản mới hơn, hoặc tính căn bậc hai của MSE để có RMSE (Root Mean Squared Error).
- **Solution:** Upgrade scikit-learn to a newer version, or calculate the square root of MSE to get RMSE.

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

# Tính MSE (Calculate MSE)
mse = mean_squared_error(y_test, y_pred)

# Tính RMSE (Calculate RMSE)
rmse = np.sqrt(mse)

print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
```

Lưu ý quan trọng (Important Notes):

- **Đường dẫn (Paths):** Đảm bảo rằng các đường dẫn đến file CSV là chính xác.
- **Kiểm tra dữ liệu (Data Checks):** Luôn kiểm tra dữ liệu sau khi đọc và tiền xử lý để đảm bảo không có lỗi.
- **Phiên bản thư viện (Library Versions):** Đảm bảo rằng bạn đang sử dụng các phiên bản thư viện tương thích.

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