PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Biểu đồ radar của các điểm giống thuốc

Tuyệt vời! Với vai trò là 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, chúng ta sẽ cùng nhau xây dựng quy trình làm việc hiệu quả và giải quyết các vấn đề phát sinh.

Tổng quan dự án và kế hoạch thực hiện:

Dự án của bạn tập trung vào việc khai thác dữ liệu từ cơ sở dữ liệu ChEMBL 35 để hỗ trợ nghiên cứu và phát triển thuốc. Quy trình làm việc tuân theo chuẩn AIMLOps Template, bao gồm:

1. **Trích xuất dữ liệu từ ChEMBL 35 bằng SQL:** Sử dụng pgAdmin để chạy các truy vấn SQL và lưu kết quả vào các file CSV trong thư mục `../data/`.
2. **Phân tích dữ liệu bằng Python và RDKit:** Sử dụng Jupyter Notebook để viết code Python, kết hợp với thư viện RDKit để xử lý và phân tích dữ liệu hóa học.
3. **Lưu trữ code và kết quả:** Lưu các file Jupyter Notebook trong thư mục `notebook/` với tên file theo định dạng `Topic_CheMBL_35_79_1_*` và `Topic_CheMBL_35_79_2_*`.

1. Phân tích mô hình (Analysis of the model):

Mục tiêu chính của bạn là khám phá các mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học của các hợp chất trong ChEMBL 35. Dưới đây là một số mô hình phân tích bạn có thể áp dụng:

- **Phân tích hoạt tính-cấu trúc (Structure-Activity Relationship - SAR):** Xác định các nhóm chức hoặc cấu trúc con (substructure) quan trọng ảnh hưởng đến hoạt tính của hợp chất.
- **Mô hình hóa định lượng hoạt tính-cấu trúc (Quantitative Structure-Activity Relationship - QSAR):** Xây dựng mô hình toán học dự đoán hoạt tính của hợp chất dựa trên các thuộc tính hóa học và vật lý của nó.
- **Phân cụm (Clustering):** Phân nhóm các hợp chất có cấu trúc tương tự nhau để tìm ra các "scaffold" (khung cấu trúc) tiềm năng.
- **Phân tích thành phần chính (Principal Component Analysis - PCA):** Giảm số chiều của dữ liệu bằng cách tìm ra các thành phần chính (principal components) thể hiện sự biến động lớn nhất trong dữ liệu.

2. Hướng dẫn song ngữ (Bilingual instructions):

Dưới đây là hướng dẫn chi tiết cho từng bước trong quy trình làm việc, kèm theo code mẫu SQL và Python:

Bước 1: Trích xuất dữ liệu từ ChEMBL 35 bằng SQL (Step 1: Extract data from ChEMBL 35 using SQL)

- **Kết nối đến cơ sở dữ liệu ChEMBL 35 (Connect to ChEMBL 35 database):**

```
-- Connect to the database using pgAdmin with the provided credentials:
-- Host: 192.168.206.136
-- User: rd
-- Password: rd
-- Database: chembl_35
```

- **Truy vấn dữ liệu và lưu vào file CSV (Query data and save to CSV file):**

Ví dụ, để lấy thông tin về các hợp chất ức chế enzyme EGFR (Epidermal Growth Factor Receptor):

```
-- Select data for compounds inhibiting EGFR
SELECT
    act.molregno,
    cmp.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    act.assay_id
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
JOIN
    assays ass ON act.assay_id = ass.assay_id
JOIN
    target_dictionary tgt ON ass.tid = tgt.tid
WHERE
    tgt.target_name = 'Epidermal Growth Factor Receptor'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
LIMIT 100;

-- Save the result to a CSV file (e.g., egfr_inhibitors.csv)
-- You can do this directly in pgAdmin by right-clicking on the query result and selecting "Copy with Headers" then pasting into a text file and saving as CSV.
```

Giải thích (Explanation):

- Câu truy vấn này kết hợp các bảng `activities`, `molecule_dictionary`, `assays`, và `target_dictionary` để lấy thông tin về hoạt tính, cấu trúc, assay và mục tiêu của các hợp chất.
- Điều kiện `tgt.target_name = 'Epidermal Growth Factor Receptor'` lọc ra các hợp chất ức chế enzyme EGFR.
- `act.standard_type = 'IC50'` và `act.standard_units = 'nM'` đảm bảo rằng chúng ta chỉ lấy các giá trị IC50 được đo bằng nM.
- `act.standard_value IS NOT NULL` loại bỏ các hàng có giá trị IC50 bị thiếu.
- `LIMIT 100` giới hạn số lượng kết quả trả về là 100 dòng.

Bước 2: Phân tích dữ liệu bằng Python và RDKit (Step 2: Analyze data using Python and RDKit)

- **Đọc dữ liệu từ file CSV (Read data from CSV file):**

```
import pandas as pd
import os

base_path = "../data"  # Đường dẫn đến thư mục chứa file CSV
file_name = "egfr_inhibitors.csv"  # Tên file CSV

file_path = os.path.join(base_path, file_name)

try:
    df = pd.read_csv(file_path)
    print(f"Đã đọc thành công file CSV từ: {file_path}")
    print(df.head())  # In ra 5 dòng đầu tiên của DataFrame
except FileNotFoundError:
```

```
    print(f"Lỗi: Không tìm thấy file CSV tại đường dẫn: {file_path}")
except Exception as e:
    print(f"Lỗi không xác định: {e}")
```

Giải thích (Explanation):

- Đoạn code này sử dụng thư viện `pandas` để đọc file CSV vào một DataFrame.
- Hàm `os.path.join()` được sử dụng để tạo đường dẫn đầy đủ đến file CSV.
- Khối `try...except` được sử dụng để xử lý các lỗi có thể xảy ra trong quá trình đọc file.

- **Sử dụng RDKit để xử lý dữ liệu hóa học (Use RDKit to process chemical data):**

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

def calculate_molecular_weight(smiles):
    """Calculates the molecular weight of a compound given its SMILES string."""
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Assuming your DataFrame 'df' has a column named 'smiles'
# Example: df['smiles'] = ['CCO', 'c1ccccc1']
# You'll need to replace this with the actual column name containing SMILES
strings in your data

# Create a sample DataFrame if 'df' does not have 'smiles' column
if 'canonical_smiles' not in df.columns:
    df['canonical_smiles'] = ['CCO', 'c1ccccc1', 'C1=CC=CC=C1C=C1C=CC=CC=1']  #
Example SMILES

# Apply the function to create a new column 'molecular_weight'
df['molecular_weight'] =
df['canonical_smiles'].apply(calculate_molecular_weight)

print(df[['canonical_smiles', 'molecular_weight']].head())
```

Giải thích (Explanation):

- Đoạn code này sử dụng thư viện `RDKit` để tính toán khối lượng phân tử của các hợp chất dựa trên chuỗi SMILES của chúng.
- Hàm `Chem.MolFromSmiles()` chuyển đổi chuỗi SMILES thành đối tượng `Mol` của RDKit.
- Hàm `Descriptors.MolWt()` tính toán khối lượng phân tử của đối tượng `Mol`.
- Hàm `apply()` được sử dụng để áp dụng hàm `calculate_molecular_weight()` cho từng hàng trong cột 'smiles' của DataFrame.

3. Giải quyết lỗi (Error handling):

- **Lỗi (Error) a:** `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`

  Lỗi này xảy ra do bạn đang cố gắng sử dụng toán tử ~ (regular expression match) trên một cột kiểu số (`numeric`). Để khắc phục, bạn cần chuyển đổi cột `act.standard_value` sang kiểu text trước khi so sánh với regular expression:

```
SELECT
    act.molregno,
    cmp.chembl_id,
```

```
    act.standard_type,
    act.standard_value,
    act.standard_units,
    act.assay_id
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
JOIN
    assays ass ON act.assay_id = ass.assay_id
JOIN
    target_dictionary tgt ON ass.tid = tgt.tid
WHERE
    tgt.target_name = 'Epidermal Growth Factor Receptor'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$' -- Convert to TEXT for
regex
LIMIT 100;
```

Explanation:

- We used `CAST(act.standard_value AS TEXT)` to explicitly convert the numeric column to text before applying the regex.

- **Lỗi (Error) b:** Phiên bản scikit-learn cũ không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`

  Nếu bạn đang sử dụng phiên bản scikit-learn cũ, hãy loại bỏ tham số `squared=False` hoặc nâng cấp lên phiên bản mới hơn:

```
from sklearn.metrics import mean_squared_error

# Nếu bạn không thể nâng cấp scikit-learn
mse = mean_squared_error(y_true, y_pred)
rmse = mse**0.5  # Tính căn bậc hai để có RMSE

# Hoặc nếu bạn có thể nâng cấp scikit-learn
# rmse = mean_squared_error(y_true, y_pred, squared=False)

print(f"Root Mean Squared Error (RMSE): {rmse}")
```

4. Ví dụ code SQL và Python mẫu (Sample SQL and Python code examples):

Dưới đây là 5 ví dụ code SQL và Python mẫu để bạn tham khảo:

Ví dụ 1: Lọc các hợp chất có khối lượng phân tử nằm trong khoảng nhất định (Filter compounds by molecular weight range)

- **SQL:**

```
-- Select compounds with molecular weight between 200 and 500
SELECT
    cmp.chembl_id,
    cmp.molecule_structures
FROM
    molecule_dictionary cmp
WHERE
    cmp.molecule_properties->>'mw_freebase' BETWEEN '200' AND '500'
LIMIT 100;
```

- **Python:**

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

base_path = "../data"
file_name = "egfr_inhibitors.csv"
file_path = os.path.join(base_path, file_name)

# Sample DataFrame, replace with your actual data loading
data = {'canonical_smiles': ['CCO', 'c1ccccc1', 'C1=CC=CC=C1C=C1C=CC=CC=1',
'CC(=O)Oc1ccccc1C(=O)O', 'C[C@H](O)c1ccccc1']}
df = pd.DataFrame(data)

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

df['molecular_weight'] =
df['canonical_smiles'].apply(calculate_molecular_weight)

# Filter compounds with molecular weight between 100 and 200
filtered_df = df[(df['molecular_weight'] >= 50) & (df['molecular_weight'] <=
200)]

print(filtered_df)
```

Ví dụ 2: Tính toán logP (Partition Coefficient) của các hợp chất (Calculate LogP of compounds)

- **SQL:** (Không thể tính logP trực tiếp bằng SQL, cần sử dụng Python)
- **Python:**

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

# Sample DataFrame, replace with your actual data loading
data = {'canonical_smiles': ['CCO', 'c1ccccc1', 'C1=CC=CC=C1C=C1C=CC=CC=1',
'CC(=O)Oc1ccccc1C(=O)O', 'C[C@H](O)c1ccccc1']}
df = pd.DataFrame(data)

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

df['logp'] = df['canonical_smiles'].apply(calculate_logp)
print(df[['canonical_smiles', 'logp']])
```

Ví dụ 3: Tìm các hợp chất tương tự với một hợp chất cho trước (Find compounds similar to a given compound)

- **SQL:** (Khó thực hiện trực tiếp bằng SQL, cần sử dụng Python và RDKit)
- **Python:**

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

# Sample DataFrame, replace with your actual data loading
data = {'canonical_smiles': ['CCO', 'c1ccccc1', 'C1=CC=CC=C1C=C1C=CC=CC=1',
'CC(=O)Oc1ccccc1C(=O)O', 'C[C@H](O)c1ccccc1']}
df = pd.DataFrame(data)

def calculate_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius=2, nBits=2048)
        return fp
    else:
        return None

df['fingerprint'] = df['canonical_smiles'].apply(calculate_fingerprint)

# Define the reference compound (query)
reference_smiles = 'c1ccccc1'
reference_fp = calculate_fingerprint(reference_smiles)

# Calculate Tanimoto similarity
def calculate_tanimoto_similarity(fp):
    if fp is None or reference_fp is None:
        return np.nan
    return Chem.DataStructs.TanimotoSimilarity(fp, reference_fp)

df['tanimoto_similarity'] =
df['fingerprint'].apply(calculate_tanimoto_similarity)

# Filter compounds with Tanimoto similarity greater than 0.5
similar_compounds = df[df['tanimoto_similarity'] > 0.5]

print(similar_compounds[['canonical_smiles', 'tanimoto_similarity']])
```

Ví dụ 4: Thống kê số lượng hợp chất cho mỗi loại hoạt tính (Count compounds for each activity type)

- **SQL:**

```sql
-- Count the number of compounds for each standard_type
SELECT
    act.standard_type,
    COUNT(DISTINCT act.molregno) AS compound_count
FROM
    activities act
WHERE
    act.standard_type IN ('IC50', 'Ki', 'EC50')
GROUP BY
    act.standard_type
ORDER BY
    compound_count DESC
LIMIT 100;
```

- **Python:**

```python
import pandas as pd

# Sample DataFrame, replace with your actual data loading
data = {'standard_type': ['IC50', 'Ki', 'IC50', 'EC50', 'Ki', 'IC50']}
df = pd.DataFrame(data)

# Count compounds for each standard type
activity_counts = df['standard_type'].value_counts().reset_index()
activity_counts.columns = ['standard_type', 'compound_count']

print(activity_counts)
```

Ví dụ 5: Tìm các mục tiêu (targets) liên quan đến một hợp chất cho trước (Find targets associated with a given compound)

- **SQL:**

```sql
-- Find targets associated with a given compound (replace 'CHEMBL123' with an
actual chembl_id)
SELECT
    tgt.target_name
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
JOIN
    assays ass ON act.assay_id = ass.assay_id
JOIN
    target_dictionary tgt ON ass.tid = tgt.tid
WHERE
    cmp.chembl_id = 'CHEMBL123'
LIMIT 100;
```

- **Python:** (Tương tự như SQL, chủ yếu là truy vấn dữ liệu)

Lưu ý quan trọng:

- Thay thế các đường dẫn file và tên cột trong code mẫu bằng các giá trị thực tế trong dự án của bạn.
- Điều chỉnh các tham số và điều kiện lọc trong các truy vấn SQL và code Python để phù hợp với mục tiêu nghiên cứu của bạn.
- Sử dụng các thư viện và công cụ phù hợp để trực quan hóa dữ liệu và khám phá các mẫu (patterns) tiềm ẩn.

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