PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Phân tích cặp phân tử giống nhau từ ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 bằng RDKit. Với kinh nghiệm trong lĩnh vực nghiên cứu và phát triển dược học, tôi sẽ cố gắng cung cấp những hướng dẫn và code mẫu tốt nhất để giúp bạn đạt được mục tiêu.

Tổng quan dự án:

Dự án của bạn tập trung vào việc khai thác dữ liệu từ cơ sở dữ liệu ChEMBL 35, kết hợp với các công cụ của RDKit để hỗ trợ quá trình nghiên cứu và phát triển thuốc. Việc sử dụng PostgreSQL (psql) để lưu trữ dữ liệu và Jupyter Notebook để phân tích là một lựa chọn hợp lý.

Cấu trúc thư mục (theo chuẩn AIMLOps Template):

Tôi hiểu rằng bạn đang tuân theo một cấu trúc thư mục chuẩn AIMLOps. Điều này giúp dự án của bạn có tổ chức và dễ dàng quản lý hơn.

Các bước thực hiện:

1. **Trích xuất dữ liệu từ PostgreSQL:** Sử dụng các truy vấn SQL để lấy dữ liệu cần thiết từ cơ sở dữ liệu ChEMBL 35. Lưu kết quả vào các file CSV trong thư mục `../data/`.
2. **Phân tích dữ liệu bằng RDKit và Jupyter Notebook:** Sử dụng RDKit để xử lý và phân tích cấu trúc hóa học, kết hợp với các thư viện Python khác (như pandas, scikit-learn) để thực hiện các phân tích thống kê và xây dựng mô hình.

Các lỗi cần chú ý:

- **Lỗi SQL:** `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`
    - Nguyên nhân: Lỗi này xảy ra do PostgreSQL không thể so sánh kiểu dữ liệu `numeric` với một chuỗi sử dụng toán tử `~` (regular expression match).
    - Cách khắc phục: Chuyển đổi cột `standard_value` sang kiểu `text` trước khi thực hiện so sánh bằng regular expression.
- **Lỗi scikit-learn:** `phiên bản 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 quá cũ và không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`.
    - Cách khắc phục: Cập nhật scikit-learn lên phiên bản mới nhất bằng lệnh `pip install -U scikit-learn`. Nếu không thể cập nhật, bạn có thể tính RMSE (Root Mean Squared Error) bằng cách lấy căn bậc hai của MSE (Mean Squared Error).

Phân tích và hướng dẫn (Song ngữ):

1. Phân tích mô hình (Model Analysis):

- **Mục tiêu (Objective):** Xác định các đặc điểm cấu trúc hóa học (chemical features) có ảnh hưởng đến hoạt tính sinh học (biological activity) của các hợp chất. Điều này có thể giúp chúng ta thiết kế các phân tử thuốc tiềm năng (potential drug molecules) hiệu quả hơn.
- **Phương pháp (Methods):**
    - **Tính toán các descriptor phân tử (Calculate molecular descriptors):** Sử dụng RDKit để tính toán các descriptor khác nhau, ví dụ: MW (Molecular Weight), LogP, số lượng H-bond donors/acceptors, Topological Polar Surface Area (TPSA),...

- **Phân tích tương quan (Correlation analysis):** Xác định mối tương quan giữa các descriptor và hoạt tính sinh học.
- **Xây dựng mô hình hồi quy (Regression model building):** Sử dụng các thuật toán học máy (ví dụ: Linear Regression, Random Forest, Support Vector Regression) để xây dựng mô hình dự đoán hoạt tính dựa trên các descriptor.
- **Đánh giá mô hình (Model evaluation):** Sử dụng các độ đo như R-squared, RMSE để đánh giá hiệu suất của mô hình.

2. Hướng dẫn (Instructions):

SQL:

- **Mục tiêu (Objective):** Trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL (Extract data from ChEMBL database).
- **Các bước (Steps):**
  1. Kết nối đến cơ sở dữ liệu PostgreSQL (Connect to PostgreSQL database).
  2. Viết các truy vấn SQL để lấy thông tin về các hợp chất và hoạt tính của chúng (Write SQL queries to retrieve information about compounds and their activities).
  3. Lưu kết quả vào file CSV (Save the results to CSV file).

Python:

- **Mục tiêu (Objective):** Phân tích dữ liệu và xây dựng mô hình dự đoán hoạt tính (Analyze data and build activity prediction models).
- **Các bước (Steps):**
  1. Đọc dữ liệu từ file CSV (Read data from CSV file).
  2. Sử dụng RDKit để xử lý cấu trúc hóa học và tính toán descriptor (Use RDKit to process chemical structures and calculate descriptors).
  3. Xây dựng và đánh giá mô hình học máy (Build and evaluate machine learning models).

3. Code mẫu (Code Examples):

SQL:

```sql
-- Ví dụ 1: Lấy 100 hợp chất và hoạt tính của chúng (Get 100 compounds and their activities)
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'
LIMIT 100;

-- Ví dụ 2: Lấy các hợp chất có hoạt tính IC50 dưới 100 nM (Get compounds with IC50 activity below 100 nM)
SELECT
    md.molregno,
    cs.canonical_smiles,
    act.standard_value
FROM
    molecule_dictionary md
```

```sql
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 IS NOT NULL
    AND act.standard_units = 'nM'
    AND act.standard_value <= 100
LIMIT 100;

-- Ví dụ 3: Tìm các hợp chất có chứa một khung nhất định (ví dụ: benzene) (Find
compounds containing a specific scaffold (e.g., benzene))
SELECT
    md.molregno,
    cs.canonical_smiles
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    cs.canonical_smiles LIKE '%c1ccccc1%'  -- Benzene ring
LIMIT 100;

-- Ví dụ 4: Sửa lỗi operator does not exist
SELECT
    md.molregno,
    cs.canonical_smiles,
    act.standard_type,
    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 = 'Ki'
    AND act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
    AND act.standard_value::text ~ '^[0-9\.]+$'  -- Convert to text for regex
LIMIT 100;

-- Ví dụ 5: Lấy số lượng hợp chất cho mỗi standard_type
SELECT act.standard_type, COUNT(*) AS compound_count
FROM activities act
GROUP BY act.standard_type
ORDER BY compound_count DESC
LIMIT 100;
```

Python:

```python
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np
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 (Base path)
base_path = "../data/"  # Thay đổi nếu cần

# Ví dụ 1: Đọc dữ liệu từ file CSV và tính toán MW (Read data from CSV and calculate
MW)
csv_file = os.path.join(base_path, "chembl_data.csv") # Replace "chembl_data.csv" with
your actual file name
df = pd.read_csv(csv_file)
df = df.head(100) # Limiting to 100 rows

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

df['MW'] = df['canonical_smiles'].apply(calculate_mw)
df = df.dropna(subset=['MW'])  # Remove rows with NaN in 'MW'
print(df.head())

# Ví dụ 2: Tính toán LogP và các descriptor khác (Calculate LogP and other
descriptors)
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.head())

# Ví dụ 3: Xây dựng mô hình hồi quy tuyến tính (Build a linear regression model)
df = df.dropna(subset=['standard_value', 'MW'])  # Remove rows with NaN
X = df[['MW']]
y = df['standard_value']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse) # Calculate RMSE
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R-squared: {r2}")

# Ví dụ 4: Sử dụng nhiều descriptor (Use multiple descriptors)
def calculate_hbd(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
```

```
        return Descriptors.NumHDonors(mol)
    else:
        return None

def calculate_hba(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.NumHAcceptors(mol)
    else:
        return None

df['HBD'] = df['canonical_smiles'].apply(calculate_hbd)
df['HBA'] = df['canonical_smiles'].apply(calculate_hba)

df = df.dropna(subset=['standard_value', 'MW', 'LogP', 'HBD', 'HBA']) # Ensure no NaNs

X = df[['MW', 'LogP', 'HBD', 'HBA']]
y = df['standard_value']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R-squared: {r2}")

# Ví dụ 5: Sử dụng Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

df = df.dropna(subset=['standard_value', 'MW', 'LogP', 'HBD', 'HBA']) # Ensure no NaNs

X = df[['MW', 'LogP', 'HBD', 'HBA']]
y = df['standard_value']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42)  # You can adjust the
number of estimators
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R-squared: {r2}")
```

Lưu ý quan trọng:

- Hãy thay thế `"chembl_data.csv"` bằng tên file CSV thực tế của bạn.
- Điều chỉnh các tham số của mô hình (ví dụ: `n_estimators` trong `RandomForestRegressor`) để đạt được hiệu suất tốt nhất.
- Việc lựa chọn descriptor phù hợp và thuật toán học máy phù hợp là rất quan trọng để xây dựng một mô hình dự đoán chính xác.

Tôi hy vọng những hướng dẫn và code mẫu này sẽ giúp bạn bắt đầu dự án của mình một cách suôn sẻ. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi! Chúc bạn thành công!