PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Phân tích so sánh thuốc hạ huyết áp

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 giúp bạn xây dựng mô hình phân tích, cung cấp hướng dẫn chi tiết bằng cả tiếng Anh và tiếng Việt, cùng với các đoạn mã SQL và Python mẫu để bạn tham khảo.

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 thư viện RDKit để hỗ trợ quá trình nghiên cứu và phát triển thuốc. Bạn đang sử dụng PostgreSQL làm hệ quản trị cơ sở dữ liệu, và tuân theo cấu trúc thư mục chuẩn AIMLOps Template.

Phân tích và Mô hình phân tích (Analysis and Analysis Model):

Mục tiêu của bạn là phân tích dữ liệu từ ChEMBL 35 để tìm ra các mối liên hệ giữa cấu trúc hóa học của các hợp chất và hoạt tính sinh học của chúng. Dưới đây là một số mô hình phân tích tiềm năng mà bạn có thể áp dụng:

1. **Phân tích tương quan cấu trúc-hoạt tính (Structure-Activity Relationship - SAR):**
    - **Mô tả:** Xác định các nhóm chức hoặc đặc điểm cấu trúc nào đóng vai trò quan trọng trong việc quyết định hoạt tính của một hợp chất.
    - **Phương pháp:** Sử dụng các mô tả cấu trúc (descriptors) từ RDKit, sau đó áp dụng các phương pháp thống kê hoặc học máy để tìm ra mối tương quan với hoạt tính sinh học.
2. **Mô hình QSAR/QSPR (Quantitative Structure-Activity/Property Relationship):**
    - **Mô tả:** Xây dựng mô hình toán học để dự đoán hoạt tính hoặc tính chất của một hợp chất dựa trên cấu trúc của nó.
    - **Phương pháp:** Sử dụng các thuật toán hồi quy (regression) hoặc phân loại (classification) để xây dựng mô hình dự đoán.
3. **Phân cụm (Clustering):**
    - **Mô tả:** Nhóm các hợp chất có cấu trúc hoặc hoạt tính tương tự nhau thành các cụm.
    - **Phương pháp:** Sử dụng các thuật toán phân cụm như k-means hoặc hierarchical clustering dựa trên các mô tả cấu trúc.
4. **Phân tích thành phần chính (Principal Component Analysis - PCA):**
    - **Mô tả:** Giảm số lượng biến (mô tả cấu trúc) bằng cách tìm ra các thành phần chính, giúp đơn giản hóa dữ liệu và trực quan hóa các mối quan hệ.
    - **Phương pháp:** Áp dụng thuật toán PCA để giảm số chiều của dữ liệu mô tả cấu trúc.

Hướng dẫn song ngữ (Bilingual Instructions):

Bước 1: Trích xuất dữ liệu từ ChEMBL 35 (Extracting Data from ChEMBL 35)

- **Tiếng Việt:** Sử dụng truy vấn SQL để lấy dữ liệu cần thiết từ cơ sở dữ liệu ChEMBL 35. Lưu kết quả vào file CSV.
- **English:** Use SQL queries to retrieve the necessary data from the ChEMBL 35 database. Save the results to a CSV file.

Bước 2: Tiền xử lý dữ liệu (Data Preprocessing)

- **Tiếng Việt:** Đọc dữ liệu từ file CSV, làm sạch và chuyển đổi dữ liệu (ví dụ: chuyển đổi SMILES sang cấu trúc phân tử bằng RDKit).
- **English:** Read data from the CSV file, clean and transform the data (e.g., convert SMILES to molecular structures using RDKit).

Bước 3: Tính toán mô tả cấu trúc (Calculating Molecular Descriptors)

- **Tiếng Việt:** Sử dụng RDKit để tính toán các mô tả cấu trúc (ví dụ: trọng lượng phân tử, logP, số lượng liên kết, v.v.).
- **English:** Use RDKit to calculate molecular descriptors (e.g., molecular weight, logP, number of bonds, etc.).

Bước 4: Xây dựng mô hình (Model Building)

- **Tiếng Việt:** Chọn một mô hình phù hợp (ví dụ: QSAR, phân cụm) và xây dựng mô hình bằng các thuật toán học máy.
- **English:** Choose an appropriate model (e.g., QSAR, clustering) and build the model using machine learning algorithms.

Bước 5: Đánh giá mô hình (Model Evaluation)

- **Tiếng Việt:** Đánh giá hiệu suất của mô hình bằng các chỉ số phù hợp (ví dụ: R-squared, RMSE, AUC).
- **English:** Evaluate the performance of the model using appropriate metrics (e.g., R-squared, RMSE, AUC).

Mã SQL và Python mẫu (Sample SQL and Python Code):

1. Trích xuất dữ liệu hoạt tính (Activity Data Extraction)

```sql
-- English
-- Extract activity data for a specific target (e.g., CHEMBL205)
-- and limit to 100 rows
SELECT
    act.molregno,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    md.chembl_id,
    md.canonical_smiles
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$' --Fix lỗi a
    AND md.chembl_id IN (SELECT DISTINCT compound_chembl_id from target_components
WHERE target_chembl_id = 'CHEMBL205')
LIMIT 100;

-- Tiếng Việt
-- Trích xuất dữ liệu hoạt tính cho một mục tiêu cụ thể (ví dụ: CHEMBL205)
-- và giới hạn kết quả 100 dòng
SELECT
    act.molregno,
    act.standard_value,
```

```
    act.standard_units,
    act.standard_type,
    md.chembl_id,
    md.canonical_smiles
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$' --Fix lỗi a
    AND md.chembl_id IN (SELECT DISTINCT compound_chembl_id from target_components
WHERE target_chembl_id = 'CHEMBL205')
LIMIT 100;
```

2. Đọc dữ liệu và tính toán mô tả cấu trúc (Data Reading and Descriptor Calculation)

```
# English
# Read CSV file and calculate molecular descriptors using RDKit
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import os

base_path = '.' # Thay đổi nếu cần

# Load data
csv_file = os.path.join(base_path, 'data', 'activity_data.csv') # Thay đổi tên file
nếu cần
df = pd.read_csv(csv_file)

# Function to calculate molecular descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        descriptors = {desc[0]: desc[1](mol) for desc in Descriptors.descList}
        return descriptors
    else:
        return None

# Apply the function to each SMILES string
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Convert descriptors to columns
df = pd.concat([df.drop('descriptors', axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

print(df.head())

# Tiếng Việt
# Đọc file CSV và tính toán các mô tả phân tử sử dụng RDKit
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import os

base_path = '.' # Thay đổi nếu cần
```

```
# Load data
csv_file = os.path.join(base_path, 'data', 'activity_data.csv') # Thay đổi tên file
nếu cần
df = pd.read_csv(csv_file)

# Hàm tính toán mô tả phân tử
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        descriptors = {desc[0]: desc[1](mol) for desc in Descriptors.descList}
        return descriptors
    else:
        return None

# Áp dụng hàm cho mỗi chuỗi SMILES
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Chuyển đổi descriptors thành các cột
df = pd.concat([df.drop('descriptors', axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

print(df.head())
```

3. Mô hình QSAR đơn giản (Simple QSAR Model)

```
# English
# Build a simple QSAR model using linear regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Prepare data
df = df.dropna()
X = df[['MolWt', 'LogP', 'NumHAcceptors', 'NumHDonors']]  # Example descriptors
y = df['standard_value']

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

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

# Predict
y_pred = model.predict(X_test)

# Evaluate
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Tiếng Việt
# Xây dựng mô hình QSAR đơn giản sử dụng hồi quy tuyến tính
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Chuẩn bị dữ liệu
df = df.dropna()
```

```
X = df[['MolWt', 'LogP', 'NumHAcceptors', 'NumHDonors']]  # Ví dụ về các descriptor
y = df['standard_value']

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

# Huấn luyện mô hình
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)

# Đánh giá
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

4. Phân cụm sử dụng K-means (Clustering using K-means)

```
# English
# Perform clustering using K-means algorithm
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Prepare data
df = df.dropna()
X = df[['MolWt', 'LogP', 'NumHAcceptors', 'NumHDonors']]  # Example descriptors

# Scale data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Cluster
kmeans = KMeans(n_clusters=3, random_state=42, n_init = 'auto') # You may need to tune
n_clusters
df['cluster'] = kmeans.fit_predict(X_scaled)

print(df[['canonical_smiles', 'cluster']].head())

# Tiếng Việt
# Thực hiện phân cụm sử dụng thuật toán K-means
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Chuẩn bị dữ liệu
df = df.dropna()
X = df[['MolWt', 'LogP', 'NumHAcceptors', 'NumHDonors']]  # Ví dụ về các descriptor

# Scale dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Phân cụm
kmeans = KMeans(n_clusters=3, random_state=42, n_init = 'auto') # Bạn có thể cần điều
chỉnh n_clusters
df['cluster'] = kmeans.fit_predict(X_scaled)

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

5. Phân tích thành phần chính (PCA)

```
# English
# Perform PCA to reduce dimensionality
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Prepare data
df = df.dropna()
X = df[['MolWt', 'LogP', 'NumHAcceptors', 'NumHDonors']]  # Example descriptors

# Scale data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# PCA
pca = PCA(n_components=2)  # Reduce to 2 components
X_pca = pca.fit_transform(X_scaled)

# Plot
plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Molecular Descriptors')
plt.show()

# Tiếng Việt
# Thực hiện PCA để giảm số chiều dữ liệu
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Chuẩn bị dữ liệu
df = df.dropna()
X = df[['MolWt', 'LogP', 'NumHAcceptors', 'NumHDonors']]  # Ví dụ về các descriptor

# Scale dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# PCA
pca = PCA(n_components=2)  # Giảm xuống còn 2 thành phần
X_pca = pca.fit_transform(X_scaled)

# Vẽ biểu đồ
plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.xlabel('Thành phần chính 1')
plt.ylabel('Thành phần chính 2')
plt.title('PCA của các mô tả phân tử')
plt.show()
```

Lưu ý về lỗi:

- **Lỗi a (SQL):** Lỗi `"operator does not exist: numeric ~ unknown"` xảy ra do bạn đang cố gắng so sánh một cột kiểu số (numeric) với một chuỗi (unknown). Để khắc phục, bạn cần ép kiểu cột `standard_value` về kiểu text trước khi sử dụng toán tử ~ (regex matching). Tôi đã sửa lỗi này trong đoạn mã SQL mẫu bằng cách sử dụng `act.standard_value::text ~ '^[0-9\.]+$'`.

- **Lỗi b (Python):** Nếu bạn gặp lỗi về tham số `squared=False` trong hàm `mean_squared_error`, hãy đảm bảo rằng bạn đang sử dụng phiên bản scikit-learn đủ mới (phiên bản 0.22 trở lên). Nếu không, bạn có thể bỏ qua tham số này, vì giá trị mặc định của `squared` là `True`, và bạn có thể tính căn bậc hai của MSE để có được RMSE.

Lưu ý quan trọng:

- Các đoạn mã trên chỉ là ví dụ minh họa. Bạn cần điều chỉnh chúng cho phù hợp với dữ liệu và mục tiêu cụ thể của bạn.
- Hãy nhớ sử dụng `os.path.join(base_path, ...)` để kết nối đường dẫn một cách an toàn và linh hoạt.
- Việc lựa chọn các mô tả cấu trúc và thuật toán học máy phù hợp là rất quan trọng để đạt được kết quả tốt.

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