PharmApp Suite

🧠 AI for Drug Discovery and Development 🧪

🧩 Topic: So sánh vị trí gắn ligand giữa các target

Tuyệt vời! Với vai trò là chuyên gia R&D 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:

- **Mục tiêu:** Phân tích dữ liệu ChEMBL 35 để hỗ trợ nghiên cứu và phát triển thuốc.
- **Dữ liệu:** ChEMBL 35 (cơ sở dữ liệu lớn chứa thông tin về các phân tử và hoạt tính sinh học của chúng).
- **Công cụ:**
    - PostgreSQL (psql) với cơ sở dữ liệu `chembl_35` (IP: 192.168.206.136, user: rd, pass: rd)
    - RDKit (toolkit hóa học tin học)
    - Jupyter Notebook
    - Scikit-learn
- **Quy trình:**
    1. Truy vấn dữ liệu từ PostgreSQL, lưu dưới dạng file CSV.
    2. Sử dụng Jupyter Notebook để phân tích dữ liệu với RDKit và Scikit-learn.
- **Cấu trúc thư mục:** Tuân theo chuẩn AIMLOps Template.
- **Yêu cầu:**
    - Code SQL và Python (tiếng Anh).
    - Giới hạn số lượng bản ghi: 100 dòng.
    - Xử lý lỗi:
        - `ERROR: operator does not exist: numeric ~ unknown`
        - `squared=False` không được hỗ trợ trong `mean_squared_error`.
- **Mã chủ đề:** Topic_CheMBL_35_30

1. Phân tích mô hình phân tích

Dựa trên mã chủ đề "Topic_CheMBL_35_30", tôi giả định rằng chủ đề này liên quan đến việc phân tích mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học của các phân tử. Một số mô hình phân tích phù hợp có thể bao gồm:

- **SAR/QSAR (Structure-Activity Relationship/Quantitative Structure-Activity Relationship):** Mô hình này cố gắng thiết lập mối quan hệ giữa cấu trúc hóa học của một phân tử và hoạt tính sinh học của nó. Chúng ta có thể sử dụng các descriptor (thuộc tính) hóa học tính toán từ RDKit để biểu diễn cấu trúc và các thuật toán học máy (ví dụ: hồi quy tuyến tính, SVM, Random Forest) để xây dựng mô hình dự đoán hoạt tính.
- **Classification:** Nếu hoạt tính sinh học được biểu diễn dưới dạng các lớp (ví dụ: active/inactive), chúng ta có thể sử dụng các thuật toán phân loại (ví dụ: Logistic Regression, Naive Bayes, SVM) để dự đoán lớp hoạt tính dựa trên cấu trúc hóa học.
- **Clustering:** Sử dụng các thuật toán clustering (ví dụ: k-means, hierarchical clustering) để nhóm các phân tử có cấu trúc và hoạt tính tương tự lại với nhau. Điều này có thể giúp chúng ta xác định các scaffold (khung) quan trọng cho hoạt tính.

Analysis Model

Based on the topic code "Topic_CheMBL_35_30," I assume that this topic relates to analyzing the relationship between the chemical structure and biological activity of molecules. Suitable analysis models may include:

- **SAR/QSAR (Structure-Activity Relationship/Quantitative Structure-Activity Relationship):** This model attempts to establish a relationship between the chemical structure of a molecule and its biological activity. We can use chemical descriptors calculated from RDKit to represent the structure and machine learning algorithms (e.g., linear regression, SVM, Random Forest) to build a model to predict activity.
- **Classification:** If biological activity is represented as classes (e.g., active/inactive), we can use classification algorithms (e.g., Logistic Regression, Naive Bayes, SVM) to predict the activity class based on the chemical structure.
- **Clustering:** Use clustering algorithms (e.g., k-means, hierarchical clustering) to group molecules with similar structures and activities together. This can help us identify important scaffolds for activity.

2. Hướng dẫn song ngữ và code mẫu

Dưới đây là hướng dẫn song ngữ và các ví dụ code SQL và Python để bạn bắt đầu.

Ví dụ 1: Trích xuất dữ liệu và tính toán descriptor

SQL (lấy 100 dòng):

```sql
-- English
SELECT
    md.molregno,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
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
LIMIT 100;

-- Vietnamese
-- Lấy thông tin về các phân tử, cấu trúc SMILES, giá trị IC50 từ cơ sở dữ liệu ChEMBL
-- Giới hạn kết quả trả về 100 dòng
SELECT
    md.molregno, -- Mã định danh phân tử
    cs.canonical_smiles, -- Cấu trúc SMILES
    act.standard_value, -- Giá trị IC50
    act.standard_units, -- Đơn vị của giá trị IC50
    act.standard_type -- Loại hoạt tính (ví dụ: IC50)
FROM
    molecule_dictionary md -- Bảng thông tin phân tử
JOIN
    compound_structures cs ON md.molregno = cs.molregno -- Bảng cấu trúc phân tử
JOIN
    activities act ON md.molregno = act.molregno -- Bảng thông tin hoạt tính
WHERE act.standard_type = 'IC50' -- Lọc theo loại hoạt tính IC50
  AND act.standard_units = 'nM' -- Lọc theo đơn vị nM
```

```
  AND act.standard_value IS NOT NULL -- Lọc các giá trị IC50 không rỗng
LIMIT 100; -- Giới hạn số lượng kết quả trả về là 100
```

Python:

```python
# English
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors

# Database credentials
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Define base path (replace with your actual base path)
base_path = '.' # Assuming the notebook is in the root of your project

# CSV file path
csv_file_path = os.path.join(base_path, 'data', 'chembl_ic50_100.csv')

# Read the CSV file
df = pd.read_csv(csv_file_path)

# Function to calculate molecular weight
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Apply the function to the 'canonical_smiles' column
df['mol_weight'] = df['canonical_smiles'].apply(calculate_mw)

# Print the first 5 rows with molecular weight
print(df.head())

# Vietnamese
import os  # Import thư viện os để làm việc với đường dẫn file
import pandas as pd  # Import thư viện pandas để làm việc với dữ liệu dạng bảng
from rdkit import Chem  # Import thư viện RDKit để xử lý thông tin hóa học
from rdkit.Chem import Descriptors  # Import các descriptor (thuộc tính) hóa học từ
RDKit

# Thông tin đăng nhập cơ sở dữ liệu
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Định nghĩa đường dẫn gốc của dự án (thay thế bằng đường dẫn thực tế của bạn)
base_path = '.' # Giả sử notebook nằm ở thư mục gốc của dự án

# Đường dẫn đến file CSV
csv_file_path = os.path.join(base_path, 'data', 'chembl_ic50_100.csv')

# Đọc file CSV vào DataFrame
```

```
df = pd.read_csv(csv_file_path)

# Hàm tính toán khối lượng phân tử
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)  # Chuyển đổi SMILES thành đối tượng phân tử của
RDKit
    if mol:
        return Descriptors.MolWt(mol)  # Tính toán khối lượng phân tử
    else:
        return None  # Trả về None nếu không thể chuyển đổi SMILES

# Áp dụng hàm tính toán khối lượng phân tử cho cột 'canonical_smiles'
df['mol_weight'] = df['canonical_smiles'].apply(calculate_mw)

# In 5 dòng đầu tiên của DataFrame, bao gồm cả khối lượng phân tử
print(df.head())
```

Ví dụ 2: Xử lý lỗi `operator does not exist: numeric ~ unknown`

Lỗi này xảy ra khi bạn cố gắng so sánh một kiểu dữ liệu số (numeric) với một kiểu dữ liệu chuỗi (unknown). Trong trường hợp này, cột `act.standard_value` có thể chứa dữ liệu không phải số. Để khắc phục, bạn có thể sử dụng hàm `regexp_match` để kiểm tra xem giá trị có phải là số hay không trước khi so sánh.

SQL:

```
-- English
SELECT
    md.molregno,
    cs.canonical_smiles,
    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
  AND act.standard_value::text ~ '^[0-9\.]+$' -- Check if the value is numeric
LIMIT 100;

-- Vietnamese
-- Lấy thông tin về các phân tử, cấu trúc SMILES, giá trị IC50 từ cơ sở dữ liệu ChEMBL
-- Chỉ lấy các giá trị IC50 là số
SELECT
    md.molregno, -- Mã định danh phân tử
    cs.canonical_smiles, -- Cấu trúc SMILES
    act.standard_value, -- Giá trị IC50
    act.standard_units -- Đơn vị của giá trị IC50
FROM
    molecule_dictionary md -- Bảng thông tin phân tử
JOIN
    compound_structures cs ON md.molregno = cs.molregno -- Bảng cấu trúc phân tử
JOIN
    activities act ON md.molregno = act.molregno -- Bảng thông tin hoạt tính
WHERE act.standard_type = 'IC50' -- Lọc theo loại hoạt tính IC50
  AND act.standard_units = 'nM' -- Lọc theo đơn vị nM
  AND act.standard_value IS NOT NULL -- Lọc các giá trị IC50 không rỗng
```

```
  AND act.standard_value::text ~ '^[0-9\.]+$' -- Kiểm tra xem giá trị có phải là số
hay không
LIMIT 100; -- Giới hạn số lượng kết quả trả về là 100
```

Ví dụ 3: Tính toán LogP và loại bỏ các phân tử không hợp lệ

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

# Database credentials
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Define base path (replace with your actual base path)
base_path = '.' # Assuming the notebook is in the root of your project

# CSV file path
csv_file_path = os.path.join(base_path, 'data', 'chembl_ic50_100.csv')

# Read the CSV file
df = pd.read_csv(csv_file_path)

# Function to calculate LogP
def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolLogP(mol)
    else:
        return None

# Apply the function to the 'canonical_smiles' column
df['logp'] = df['canonical_smiles'].apply(calculate_logp)

# Remove rows where LogP is None (invalid molecules)
df = df.dropna(subset=['logp'])

# Print the first 5 rows with LogP
print(df.head())

# Vietnamese
import os  # Import thư viện os để làm việc với đường dẫn file
import pandas as pd  # Import thư viện pandas để làm việc với dữ liệu dạng bảng
from rdkit import Chem  # Import thư viện RDKit để xử lý thông tin hóa học
from rdkit.Chem import Descriptors  # Import các descriptor (thuộc tính) hóa học từ
RDKit

# Thông tin đăng nhập cơ sở dữ liệu
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Định nghĩa đường dẫn gốc của dự án (thay thế bằng đường dẫn thực tế của bạn)
base_path = '.' # Giả sử notebook nằm ở thư mục gốc của dự án
```

```
# Đường dẫn đến file CSV
csv_file_path = os.path.join(base_path, 'data', 'chembl_ic50_100.csv')

# Đọc file CSV vào DataFrame
df = pd.read_csv(csv_file_path)

# Hàm tính toán LogP
def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)  # Chuyển đổi SMILES thành đối tượng phân tử của
RDKit
    if mol:
        return Descriptors.MolLogP(mol)  # Tính toán LogP
    else:
        return None  # Trả về None nếu không thể chuyển đổi SMILES

# Áp dụng hàm tính toán LogP cho cột 'canonical_smiles'
df['logp'] = df['canonical_smiles'].apply(calculate_logp)

# Loại bỏ các hàng có giá trị LogP là None (các phân tử không hợp lệ)
df = df.dropna(subset=['logp'])

# In 5 dòng đầu tiên của DataFrame, bao gồm cả LogP
print(df.head())
```

Ví dụ 4: Chuẩn bị dữ liệu cho mô hình QSAR

```
# English
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np

# Database credentials
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Define base path (replace with your actual base path)
base_path = '.' # Assuming the notebook is in the root of your project

# CSV file path
csv_file_path = os.path.join(base_path, 'data', 'chembl_ic50_100.csv')

# Read the CSV file
df = pd.read_csv(csv_file_path)

# Function to calculate molecular descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol)]  # Example
descriptors
    else:
        return None

# Apply the function to the 'canonical_smiles' column
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)
```

```
# Remove rows where descriptors is None (invalid molecules)
df = df.dropna(subset=['descriptors'])

# Convert descriptors to separate columns
df[['mol_weight', 'logp']] = pd.DataFrame(df['descriptors'].tolist(), index=df.index)

# Convert IC50 to pIC50
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Convert nM to M

# Print the first 5 rows with descriptors and pIC50
print(df.head())

# Vietnamese
import os  # Import thư viện os để làm việc với đường dẫn file
import pandas as pd  # Import thư viện pandas để làm việc với dữ liệu dạng bảng
from rdkit import Chem  # Import thư viện RDKit để xử lý thông tin hóa học
from rdkit.Chem import Descriptors  # Import các descriptor (thuộc tính) hóa học từ
RDKit
import numpy as np # Import thư viện numpy để tính toán số học

# Thông tin đăng nhập cơ sở dữ liệu
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Định nghĩa đường dẫn gốc của dự án (thay thế bằng đường dẫn thực tế của bạn)
base_path = '.' # Giả sử notebook nằm ở thư mục gốc của dự án

# Đường dẫn đến file CSV
csv_file_path = os.path.join(base_path, 'data', 'chembl_ic50_100.csv')

# Đọc file CSV vào DataFrame
df = pd.read_csv(csv_file_path)

# Hàm tính toán các descriptor phân tử
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)  # Chuyển đổi SMILES thành đối tượng phân tử của
RDKit
    if mol:
        return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol)]  # Ví dụ: Khối lượng
phân tử và LogP
    else:
        return None  # Trả về None nếu không thể chuyển đổi SMILES

# Áp dụng hàm tính toán descriptor cho cột 'canonical_smiles'
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Loại bỏ các hàng có giá trị descriptor là None (các phân tử không hợp lệ)
df = df.dropna(subset=['descriptors'])

# Chuyển đổi các descriptor thành các cột riêng biệt
df[['mol_weight', 'logp']] = pd.DataFrame(df['descriptors'].tolist(), index=df.index)

# Chuyển đổi IC50 thành pIC50
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Chuyển đổi nM thành M

# In 5 dòng đầu tiên của DataFrame, bao gồm các descriptor và pIC50
print(df.head())
```

Ví dụ 5: Xây dựng mô hình hồi quy tuyến tính đơn giản

```
# English
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

# Database credentials
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Define base path (replace with your actual base path)
base_path = '.' # Assuming the notebook is in the root of your project

# CSV file path
csv_file_path = os.path.join(base_path, 'data', 'chembl_ic50_100.csv')

# Read the CSV file
df = pd.read_csv(csv_file_path)

# Function to calculate molecular descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol)]  # Example
descriptors
    else:
        return None

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

# Remove rows where descriptors is None (invalid molecules)
df = df.dropna(subset=['descriptors'])

# Convert descriptors to separate columns
df[['mol_weight', 'logp']] = pd.DataFrame(df['descriptors'].tolist(), index=df.index)

# Convert IC50 to pIC50
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Convert nM to M

# Prepare data for modeling
X = df[['mol_weight', 'logp']]
y = df['pIC50']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train the model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Make predictions
y_pred = model.predict(X_test)

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

# Vietnamese
import os  # Import thư viện os để làm việc với đường dẫn file
import pandas as pd  # Import thư viện pandas để làm việc với dữ liệu dạng bảng
from rdkit import Chem  # Import thư viện RDKit để xử lý thông tin hóa học
from rdkit.Chem import Descriptors  # Import các descriptor (thuộc tính) hóa học từ
RDKit
import numpy as np # Import thư viện numpy để tính toán số học
from sklearn.model_selection import train_test_split  # Import hàm chia dữ liệu
from sklearn.linear_model import LinearRegression  # Import mô hình hồi quy tuyến tính
from sklearn.metrics import mean_squared_error  # Import hàm đánh giá mô hình

# Thông tin đăng nhập cơ sở dữ liệu
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Định nghĩa đường dẫn gốc của dự án (thay thế bằng đường dẫn thực tế của bạn)
base_path = '.' # Giả sử notebook nằm ở thư mục gốc của dự án

# Đường dẫn đến file CSV
csv_file_path = os.path.join(base_path, 'data', 'chembl_ic50_100.csv')

# Đọc file CSV vào DataFrame
df = pd.read_csv(csv_file_path)

# Hàm tính toán các descriptor phân tử
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)  # Chuyển đổi SMILES thành đối tượng phân tử của
RDKit
    if mol:
        return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol)]  # Ví dụ: Khối lượng
phân tử và LogP
    else:
        return None  # Trả về None nếu không thể chuyển đổi SMILES

# Áp dụng hàm tính toán descriptor cho cột 'canonical_smiles'
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Loại bỏ các hàng có giá trị descriptor là None (các phân tử không hợp lệ)
df = df.dropna(subset=['descriptors'])

# Chuyển đổi các descriptor thành các cột riêng biệt
df[['mol_weight', 'logp']] = pd.DataFrame(df['descriptors'].tolist(), index=df.index)

# Chuyển đổi IC50 thành pIC50
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Chuyển đổi nM thành M

# Chuẩn bị dữ liệu cho mô hình
X = df[['mol_weight', 'logp']]  # Các descriptor làm biến độc lập
y = df['pIC50']  # pIC50 làm biến phụ thuộc
```

```
# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Tạo và huấn luyện mô hình
model = LinearRegression()  # Khởi tạo mô hình hồi quy tuyến tính
model.fit(X_train, y_train)  # Huấn luyện mô hình trên tập huấn luyện

# Dự đoán giá trị
y_pred = model.predict(X_test)  # Dự đoán trên tập kiểm tra

# Đánh giá mô hình
mse = mean_squared_error(y_test, y_pred)  # Tính Mean Squared Error
print(f"Mean Squared Error: {mse}")  # In kết quả
```

Xử lý lỗi `squared=False` trong `mean_squared_error`:

Nếu bạn đang sử dụng phiên bản scikit-learn cũ, tham số `squared=False` có thể không được hỗ trợ. Trong trường hợp này, bạn có thể tính căn bậc hai của MSE để có được RMSE (Root Mean Squared Error):

```
# English
mse = mean_squared_error(y_test, y_pred)
rmse = mse**0.5
print(f"Root Mean Squared Error: {rmse}")

# Vietnamese
mse = mean_squared_error(y_test, y_pred)  # Tính Mean Squared Error
rmse = mse**0.5  # Tính Root Mean Squared Error bằng cách lấy căn bậc hai của MSE
print(f"Root Mean Squared Error: {rmse}")  # In kết quả
```

Lưu ý quan trọng:

- Hãy nhớ thay thế các đường dẫn file và thông tin đăng nhập cơ sở dữ liệu bằng thông tin thực tế của bạn.
- Các ví dụ trên chỉ là điểm khởi đầu. Bạn có thể tùy chỉnh và mở rộng chúng để phù hợp với nhu cầu cụ thể của dự án.
- Việc lựa chọn descriptor và thuật toán học máy phù hợp là rất quan trọng để xây dựng các mô hình QSAR/SAR hiệu quả. Hãy thử nghiệm với các lựa chọn khác nhau và đánh giá hiệu suất của chúng một cách cẩn thận.

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