PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Tìm các phân tử giống sản phẩm tự nhiên

Tuyệt vời! Tôi sẽ vào vai một Chuyên gia R&D Dược học và giúp bạn phân tích, hướng dẫn chi tiết dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit này. Với kinh nghiệm của mình, tôi sẽ tập trung vào việc xây dựng mô hình rõ ràng, cung cấp hướng dẫn song ngữ dễ hiểu, code mẫu SQL và Python hiệu quả, đồng thời giải quyết các lỗi bạn gặp phải.

Tổng quan dự án:

Dự án của bạn tập trung vào việc tận dụng dữ liệu ChEMBL 35 và RDKit để hỗ trợ nghiên cứu và phát triển thuốc. Việc sử dụng cơ sở dữ liệu PostgreSQL (psql) với thông tin bạn cung cấp (IP, user, password, database) là một lựa chọn tốt để quản lý và truy vấn dữ liệu lớn từ ChEMBL. Việc tuân thủ AIMLOps Template giúp bạn tổ chức code và dữ liệu một cách khoa học, dễ quản lý và tái sử dụng.

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

Mục tiêu chính của bạn là khám phá và trích xuất thông tin giá trị từ dữ liệu ChEMBL để hỗ trợ quá trình phát triển thuốc. Dưới đây là một số mô hình phân tích bạn có thể áp dụng:

- **Phân tích thống kê mô tả (Descriptive Statistical Analysis):**
  - Tính toán các thống kê mô tả (mean, median, standard deviation, min, max, percentiles) cho các thuộc tính hóa học và hoạt tính sinh học.
  - Vẽ biểu đồ phân phối (histograms, box plots) để hiểu rõ hơn về dữ liệu.
  - Xác định các xu hướng và bất thường trong dữ liệu.
- **Phân tích tương quan (Correlation Analysis):**
  - Tìm kiếm mối tương quan giữa các thuộc tính hóa học (ví dụ: LogP, Molecular Weight) và hoạt tính sinh học (ví dụ: IC50, Ki).
  - Sử dụng ma trận tương quan (correlation matrix) và heatmap để trực quan hóa các mối tương quan.
- **Phân tích cấu trúc-hoạt tính (Structure-Activity Relationship - SAR Analysis):**
  - Xác định các nhóm chức (functional groups) hoặc khung phân tử (molecular scaffolds) quan trọng ảnh hưởng đến hoạt tính sinh học.
  - Sử dụng RDKit để tính toán các descriptor phân tử (molecular descriptors) và xây dựng mô hình SAR.
- **Mô hình hóa QSAR/QSPR (Quantitative Structure-Activity/Property Relationship):**
  - Xây dựng mô hình dự đoán hoạt tính sinh học hoặc tính chất của các hợp chất dựa trên cấu trúc hóa học của chúng.
  - Sử dụng các thuật toán học máy (machine learning) như Linear Regression, Random Forest, Support Vector Machines.
- **Phân tích cụm (Clustering Analysis):**
  - Phân nhóm các hợp chất có cấu trúc hoặc hoạt tính tương tự nhau.
  - Sử dụng các thuật toán clustering như k-means, hierarchical clustering.

2. Hướng dẫn song ngữ (Bilingual Guidance):

- **Kết nối đến cơ sở dữ liệu ChEMBL (Connecting to the ChEMBL database):**
  - **Tiếng Anh:** Use `psycopg2` library to connect to your PostgreSQL database.

- **Tiếng Việt:** Sử dụng thư viện `psycopg2` để kết nối đến cơ sở dữ liệu PostgreSQL của bạn.

- **Truy vấn dữ liệu từ ChEMBL (Querying data from ChEMBL):**
    - **Tiếng Anh:** Write SQL queries to extract relevant data from ChEMBL tables (e.g., `activities`, `molecule_dictionary`).
    - **Tiếng Việt:** Viết các truy vấn SQL để trích xuất dữ liệu liên quan từ các bảng ChEMBL (ví dụ: `activities`, `molecule_dictionary`).
- **Tính toán descriptor phân tử bằng RDKit (Calculating molecular descriptors using RDKit):**
    - **Tiếng Anh:** Use RDKit functions to calculate molecular descriptors (e.g., MolWt, LogP, TPSA) from SMILES strings.
    - **Tiếng Việt:** Sử dụng các hàm của RDKit để tính toán các descriptor phân tử (ví dụ: MolWt, LogP, TPSA) từ chuỗi SMILES.
- **Xây dựng mô hình học máy (Building machine learning models):**
    - **Tiếng Anh:** Use scikit-learn to build and evaluate machine learning models for QSAR/QSPR.
    - **Tiếng Việt:** Sử dụng scikit-learn để xây dựng và đánh giá các mô hình học máy cho QSAR/QSPR.

3. Code SQL, Python mẫu (Sample SQL and Python Code):

SQL (để chạy trên pgAdmin và lưu vào file .csv):

```sql
-- Lấy 100 dòng dữ liệu hoạt tính sinh học (IC50) cho một target cụ thể (ví dụ: EGFR)
-- Get 100 rows of bioactivity data (IC50) for a specific target (e.g., EGFR)
SELECT
    md.chembl_id,
    md.molecule_structures,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Epidermal Growth Factor Receptor'  -- Thay đổi target nếu cần / Change target if needed
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0  -- Loại bỏ các giá trị không hợp lệ / Remove invalid values
LIMIT 100;
```

Lưu ý về lỗi SQL:

Lỗi `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'` xảy ra do PostgreSQL không thể sử dụng toán tử `~` (regex match) trực tiếp trên cột kiểu `numeric`. Để khắc phục, bạn có thể chuyển đổi cột `standard_value` sang kiểu `text` trước khi so sánh:

```sql
-- Sửa lỗi bằng cách chuyển đổi sang text
-- Fix the error by casting to text
SELECT
```

```
    md.chembl_id,
    md.molecule_structures,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Epidermal Growth Factor Receptor'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'  -- Chuyển đổi sang text /
Convert to text
LIMIT 100;
```

Python (trong Jupyter Notebook):

```
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import psycopg2
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Cấu hình kết nối database / Database connection configuration
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Đường dẫn cơ sở của dự án / Base path of the project
base_path = '.'  # Thay đổi nếu cần / Change if needed
data_path = os.path.join(base_path, 'data')
notebook_path = os.path.join(base_path, 'notebooks')

# Đảm bảo thư mục tồn tại / Ensure directories exist
os.makedirs(data_path, exist_ok=True)
os.makedirs(notebook_path, exist_ok=True)

# Tên file CSV / CSV filename
csv_file = os.path.join(data_path, 'chembl_egfr_ic50_100.csv') #Đặt tên file sau khi
chạy SQL và export

# Hàm kết nối đến database / Function to connect to the database
def connect_to_db(ip, user, password, database):
    conn = psycopg2.connect(host=ip, user=user, password=password, database=database)
    return conn

# Hàm tính toán descriptor phân tử / Function to calculate molecular descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
```

```
        return None
    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    descriptors['HBD'] = Descriptors.NumHDonors(mol)
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
    return descriptors

# Kết nối đến database / Connect to the database
conn = connect_to_db(db_ip, db_user, db_pass, db_name)

# Đọc dữ liệu từ file CSV (sau khi đã export từ pgAdmin) / Read data from CSV file
(after exporting from pgAdmin)
try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}.  Please ensure the file exists.")
    exit()

# In ra thông tin cơ bản về DataFrame / Print basic information about the DataFrame
print("DataFrame Info:")
print(df.info())

# In ra một vài dòng đầu tiên của DataFrame / Print the first few rows of the
DataFrame
print("\nFirst 5 rows of the DataFrame:")
print(df.head())


# Tiền xử lý dữ liệu: Chuyển đổi IC50 sang pIC50 / Data preprocessing: Convert IC50 to
pIC50
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9) # Chuyển đổi nM sang M / Convert
nM to M

# Áp dụng tính toán descriptor và thêm vào DataFrame / Apply descriptor calculation
and add to DataFrame
df['descriptors'] = df['molecule_structures'].apply(calculate_descriptors)

# Loại bỏ các hàng có descriptor bị lỗi / Remove rows with missing descriptors
df = df.dropna(subset=['descriptors'])

# Chuyển đổi descriptor thành các cột riêng biệt / Convert descriptors to separate
columns
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

# Loại bỏ các hàng có giá trị NaN sau khi tính toán descriptor / Remove rows with NaN
values after descriptor calculation
df = df.dropna()

# Chọn các features và target / Select features and target
features = ['MolWt', 'LogP', 'HBD', 'HBA']
target = 'pIC50'

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra / Split data into training and
testing sets
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target],
test_size=0.2, random_state=42)
```

```python
# Xây dựng mô hình Linear Regression / Build a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra / Make predictions on the test set
y_pred = model.predict(X_test)

# Đánh giá mô hình / Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

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

Lưu ý về lỗi Python:

- **Lỗi `ValueError: squared is not a valid keyword for this function`:** Lỗi này xảy ra khi bạn sử dụng phiên bản scikit-learn cũ không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`. Để khắc phục, hãy nâng cấp scikit-learn lên phiên bản mới nhất bằng lệnh: `pip install -U scikit-learn`. Nếu không thể nâng cấp, bạn có thể tính RMSE (Root Mean Squared Error) bằng cách lấy căn bậc hai của MSE: `rmse = np.sqrt(mse)`.

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

Ví dụ 1: SQL - Lấy thông tin về các hợp chất có hoạt tính cao trên một target cụ thể:

```sql
-- Lấy thông tin về các hợp chất có IC50 < 100 nM trên target EGFR
-- Get information about compounds with IC50 < 100 nM on target EGFR
SELECT
    md.chembl_id,
    md.molecule_structures,
    act.standard_value
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Epidermal Growth Factor Receptor'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value < 100
LIMIT 100;
```

Ví dụ 1: Python - Tính toán descriptor và vẽ biểu đồ:

```python
import pandas as pd
import matplotlib.pyplot as plt
from rdkit import Chem
from rdkit.Chem import Descriptors

def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    return descriptors
```

```
df = pd.read_csv('chembl_egfr_ic50_100.csv')
df['descriptors'] = df['molecule_structures'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors'])
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

plt.scatter(df['MolWt'], df['LogP'])
plt.xlabel('Molecular Weight')
plt.ylabel('LogP')
plt.title('Molecular Weight vs LogP')
plt.show()
```

Ví dụ 2: SQL - Lấy thông tin về các target liên quan đến một bệnh cụ thể (ví dụ: ung thư):

```
-- Lấy thông tin về các target liên quan đến bệnh ung thư
-- Get information about targets related to cancer
SELECT
    td.tid,
    td.pref_name,
    td.target_type
FROM
    target_dictionary td
WHERE
    td.organism = 'Homo sapiens'
    AND td.target_class LIKE '%Kinase%' -- Tìm kiếm các kinase liên quan đến ung thư /
Search for kinases related to cancer
LIMIT 100;
```

Ví dụ 2: Python - Xây dựng mô hình QSAR đơn giản:

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    return descriptors

df = pd.read_csv('chembl_egfr_ic50_100.csv')
df['descriptors'] = df['molecule_structures'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors'])
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

X = df[['MolWt', 'LogP']]
y = -np.log10(df['standard_value'] * 1e-9) # pIC50

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)
print(f'Mean Squared Error: {mse}')
```

Ví dụ 3: SQL - Tìm kiếm các hợp chất tương tự về cấu trúc:

```
-- (Cần cài đặt extension pg_trgm trong PostgreSQL)
-- (Requires pg_trgm extension in PostgreSQL)
CREATE EXTENSION IF NOT EXISTS pg_trgm;

SELECT
    md1.chembl_id AS chembl_id1,
    md2.chembl_id AS chembl_id2
FROM
    molecule_dictionary md1,
    molecule_dictionary md2
WHERE
    md1.molregno != md2.molregno
    AND md1.molecule_structures % md2.molecule_structures -- Sử dụng toán tử % cho sự tương đồng chuỗi / Use % operator for string similarity
LIMIT 100;
```

Ví dụ 3: Python - Sử dụng RDKit để tính toán fingerprint và so sánh độ tương đồng:

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

def calculate_similarity(smiles1, smiles2):
    mol1 = Chem.MolFromSmiles(smiles1)
    mol2 = Chem.MolFromSmiles(smiles2)
    if mol1 is None or mol2 is None:
        return None
    fp1 = AllChem.GetMorganFingerprintAsBitVect(mol1, 2, nBits=1024)
    fp2 = AllChem.GetMorganFingerprintAsBitVect(mol2, 2, nBits=1024)
    return DataStructs.TanimotoSimilarity(fp1, fp2)

df = pd.read_csv('chembl_egfr_ic50_100.csv')
smiles1 = df['molecule_structures'][0]
smiles2 = df['molecule_structures'][1]

similarity = calculate_similarity(smiles1, smiles2)
print(f'Tanimoto Similarity: {similarity}')
```

Ví dụ 4: SQL - Thống kê số lượng hợp chất cho mỗi target:

```
-- Thống kê số lượng hợp chất cho mỗi target
-- Count the number of compounds for each target
SELECT
    td.pref_name,
    COUNT(DISTINCT act.molregno) AS num_compounds
FROM
    activities act
JOIN
    target_dictionary td ON act.tid = td.tid
GROUP BY
    td.pref_name
ORDER BY
    num_compounds DESC
LIMIT 100;
```

Ví dụ 4: Python - Phân tích tương quan giữa các descriptor:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('chembl_egfr_ic50_100.csv')
# Đảm bảo rằng các descriptor đã được tính toán và có trong DataFrame
# Ensure that descriptors have been calculated and are in the DataFrame
if 'MolWt' in df.columns and 'LogP' in df.columns and 'HBD' in df.columns and 'HBA' in
df.columns:
    correlation_matrix = df[['MolWt', 'LogP', 'HBD', 'HBA']].corr()
    sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
    plt.title('Correlation Matrix of Molecular Descriptors')
    plt.show()
else:
    print("Error: Molecular descriptors not found in the DataFrame. Please calculate
them first.")
```

Ví dụ 5: SQL - Lọc các hợp chất dựa trên khoảng giá trị LogP:

```sql
-- Lọc các hợp chất có LogP nằm trong khoảng từ 2 đến 4
-- Filter compounds with LogP between 2 and 4
SELECT
    md.chembl_id,
    md.molecule_structures
FROM
    molecule_dictionary md
WHERE
    md.molecule_properties->>'ALOGP' BETWEEN '2' AND '4'
LIMIT 100;
```

Ví dụ 5: Python - Sử dụng scikit-learn để xây dựng mô hình Random Forest:

```python
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    return descriptors

df = pd.read_csv('chembl_egfr_ic50_100.csv')
df['descriptors'] = df['molecule_structures'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors'])
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

X = df[['MolWt', 'LogP']]
y = -np.log10(df['standard_value'] * 1e-9) # pIC50

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)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

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

Lưu ý quan trọng:

- **Data Cleaning:** Luôn luôn kiểm tra và làm sạch dữ liệu trước khi phân tích. Loại bỏ các giá trị null, duplicate, hoặc các giá trị không hợp lệ.
- **Feature Engineering:** Thử nghiệm với các descriptor phân tử khác nhau và kết hợp chúng để cải thiện hiệu suất mô hình.
- **Model Selection:** So sánh hiệu suất của các thuật toán học máy khác nhau để tìm ra mô hình tốt nhất cho bài toán của bạn.
- **Cross-Validation:** Sử dụng cross-validation để đánh giá mô hình một cách khách quan và tránh overfitting.

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