PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Ca lâm sàng: chất ức chế JAK từ ChEMBL

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 hiểu tầm quan trọng của việc khai thác dữ liệu lớn để tìm kiếm các ứng cử viên thuốc tiềm năng.

Dưới đây là phân tích chi tiết, hướng dẫn song ngữ và các ví dụ code SQL và Python (kèm theo xử lý các lỗi bạn đã đề cập) để bạn bắt đầu dự án của mình.

1. Phân tích Mô hình (Analysis Model)

Chúng ta sẽ tập trung vào việc khai thác dữ liệu từ ChEMBL 35 để xây dựng các mô hình dự đoán hoạt tính sinh học của các hợp chất. Quá trình này bao gồm các bước sau:

- **Data Extraction (Trích xuất dữ liệu):** Sử dụng SQL để truy vấn và trích xuất dữ liệu liên quan từ database ChEMBL 35.
- **Data Preprocessing (Tiền xử lý dữ liệu):** Làm sạch và chuẩn hóa dữ liệu, bao gồm xử lý giá trị thiếu, loại bỏ các giá trị không hợp lệ và chuyển đổi dữ liệu về định dạng phù hợp.
- **Feature Engineering (Xây dựng đặc trưng):** Sử dụng RDKit để tính toán các đặc trưng hóa học (ví dụ: trọng lượng phân tử, logP, số lượng liên kết quay) từ cấu trúc phân tử của các hợp chất.
- **Model Building (Xây dựng mô hình):** Sử dụng các thuật toán học máy (ví dụ: hồi quy tuyến tính, Support Vector Machines, Random Forests) để xây dựng mô hình dự đoán hoạt tính sinh học dựa trên các đặc trưng hóa học.
- **Model Evaluation (Đánh giá mô hình):** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp (ví dụ: R-squared, RMSE, AUC).

2. Hướng dẫn Song ngữ (Bilingual Guidance)

2.1. SQL (Structured Query Language)

- **Purpose (Mục đích):** Used to query and extract data from the ChEMBL 35 database. (Sử dụng để truy vấn và trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL 35.)
- **Key Concepts (Các khái niệm chính):**
    - `SELECT`: Used to specify the columns to retrieve. (Sử dụng để chỉ định các cột cần truy xuất.)
    - `FROM`: Used to specify the table to retrieve data from. (Sử dụng để chỉ định bảng cần truy xuất dữ liệu.)
    - `WHERE`: Used to filter data based on specific conditions. (Sử dụng để lọc dữ liệu dựa trên các điều kiện cụ thể.)
    - `JOIN`: Used to combine data from multiple tables. (Sử dụng để kết hợp dữ liệu từ nhiều bảng.)
    - `LIMIT`: Limits the number of rows returned. (Giới hạn số lượng hàng trả về.)

2.2. Python

- **Purpose (Mục đích):** Used for data preprocessing, feature engineering, model building, and model evaluation. (Sử dụng để tiền xử lý dữ liệu, xây dựng đặc trưng, xây dựng mô hình và đánh giá mô hình.)
- **Key Libraries (Các thư viện chính):**

- `pandas`: Used for data manipulation and analysis. (Sử dụng để thao tác và phân tích dữ liệu.)
- `RDKit`: Used for cheminformatics tasks, such as calculating molecular properties. (Sử dụng cho các tác vụ tin học hóa học, chẳng hạn như tính toán các thuộc tính phân tử.)
- `scikit-learn`: Used for machine learning tasks, such as model building and evaluation. (Sử dụng cho các tác vụ học máy, chẳng hạn như xây dựng và đánh giá mô hình.)

3. Ví dụ Code (Code Examples)

3.1. SQL

```sql
-- Get 100 compounds with IC50 values for a specific target (e.g., target_chembl_id = 'CHEMBL205')
-- Lấy 100 hợp chất với giá trị IC50 cho một mục tiêu cụ thể (ví dụ: target_chembl_id = 'CHEMBL205')
SELECT
    md.chembl_id,
    act.standard_value,
    act.standard_units
FROM
    molecule_dictionary md
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.target_chembl_id = 'CHEMBL205'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$' -- Fix for the numeric ~ unknown error
LIMIT 100;
```

Lưu ý:

- Tôi đã sửa lỗi `numeric ~ unknown` bằng cách ép kiểu `act.standard_value` về kiểu `text` trước khi so sánh với biểu thức chính quy. This ensures compatibility for the comparison.
- Câu truy vấn này lấy thông tin `chembl_id` từ bảng `molecule_dictionary`, `standard_value` và `standard_units` từ bảng `activities`. Nó kết hợp các bảng dựa trên `molregno` và `tid`, lọc kết quả cho một mục tiêu cụ thể, loại hoạt tính ('IC50'), đơn vị ('nM') và giới hạn số lượng kết quả trả về là 100.

3.2. Python

```python
import os
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, r2_score
import numpy as np

# Define base path
base_path = "."  # Assuming the notebook is in the root directory

# Load data from CSV (replace 'your_data.csv' with the actual file name)
data_path = os.path.join(base_path, 'data', 'chembl_data.csv')
try:
    df = pd.read_csv(data_path)
```

```
except FileNotFoundError:
    print(f"Error: File not found at {data_path}. Please make sure the file exists.")
    exit()


# Data Cleaning and Preprocessing
df.dropna(subset=['standard_value'], inplace=True)  # Remove rows with missing IC50
values
df = df[pd.to_numeric(df['standard_value'], errors='coerce').notna()] #remove non
numeric values
df['standard_value'] = pd.to_numeric(df['standard_value']) # Convert IC50 to numeric

# Function to calculate molecular descriptors using RDKit
def calculate_descriptors(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None
        descriptors = {desc_name: desc_func(mol) for desc_name, desc_func in
Descriptors.descList}
        return descriptors
    except:
        return None

# Apply descriptor calculation to each molecule (assuming you have a 'smiles' or
'canonical_smiles' column)
# Assuming you have a 'canonical_smiles' column
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Handle cases where descriptor calculation failed
df = df.dropna(subset=['descriptors'])

# Convert descriptors to a DataFrame and merge with the main DataFrame
descriptors_df = pd.DataFrame(df['descriptors'].tolist())
df = pd.concat([df, descriptors_df], axis=1)

# Prepare data for machine learning
X = df.drop(['chembl_id', 'standard_value', 'descriptors', 'canonical_smiles'],
axis=1, errors='ignore') #drop canonical_smiles
y = df['standard_value']

# Handle missing values (imputation)
X = X.replace([np.inf, -np.inf], np.nan) # Replace infinite values with NaN
X = X.fillna(X.mean())  # Replace NaN with the mean of each column

# 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)

# Train a linear regression 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)
r2 = r2_score(y_test, y_pred)
```

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

Lưu ý:

- Đảm bảo bạn đã cài đặt tất cả các thư viện cần thiết (pandas, RDKit, scikit-learn).
- Thay thế `'your_data.csv'` bằng tên file CSV thực tế của bạn.
- Đoạn code này giả định rằng bạn có một cột `'canonical_smiles'` chứa cấu trúc SMILES của các hợp chất. If your column is named differently, adjust the code accordingly.
- Tôi đã thêm xử lý lỗi để đảm bảo rằng chương trình không bị crash nếu gặp lỗi khi tính toán các đặc trưng hóa học.
- Tôi đã thêm xử lý giá trị vô hạn (inf) và giá trị thiếu (NaN) trong dữ liệu.
- Tôi bỏ `canonical_smiles` vì nó không cần thiết để huấn luyện mô hình.
- Đoạn code này chỉ là một ví dụ đơn giản. Bạn có thể tùy chỉnh nó để sử dụng các thuật toán học máy khác nhau và đánh giá mô hình bằng các chỉ số khác nhau.

4. Năm ví dụ SQL và Python (Five SQL and Python Examples)

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

Ví dụ 1: Trích xuất thông tin cơ bản về hợp chất (Extract Basic Compound Information)

SQL:

```sql
-- Get the ChEMBL ID and canonical SMILES for the first 100 compounds
-- Lấy ChEMBL ID và canonical SMILES cho 100 hợp chất đầu tiên
SELECT chembl_id, canonical_smiles
FROM molecule_dictionary
LIMIT 100;
```

Python:

```python
import pandas as pd
import psycopg2

# Database connection details
db_host = "192.168.206.136"
db_name = "chembl_35"
db_user = "rd"
db_pass = "rd"
db_port = "5432"

# Establish database connection
conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass, port=db_port)

# SQL query
sql_query = "SELECT chembl_id, canonical_smiles FROM molecule_dictionary LIMIT 100;"

# Read data into a pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Close database connection
conn.close()

# Print the DataFrame
print(df.head())
```

Ví dụ 2: Lọc các hợp chất theo trọng lượng phân tử (Filter Compounds by Molecular Weight)

SQL:

```sql
-- Get compounds with molecular weight between 400 and 500
-- Lấy các hợp chất có trọng lượng phân tử từ 400 đến 500
SELECT md.chembl_id, r.mw_freebase
FROM molecule_dictionary md
JOIN molecule_properties r ON md.molregno = r.molregno
WHERE r.mw_freebase BETWEEN 400 AND 500
LIMIT 100;
```

Python:

```python
import pandas as pd
import psycopg2

# Database connection details
db_host = "192.168.206.136"
db_name = "chembl_35"
db_user = "rd"
db_pass = "rd"
db_port = "5432"

# Establish database connection
conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass, port=db_port)

# SQL query
sql_query = """
SELECT md.chembl_id, r.mw_freebase
FROM molecule_dictionary md
JOIN molecule_properties r ON md.molregno = r.molregno
WHERE r.mw_freebase BETWEEN 400 AND 500
LIMIT 100;
"""

# Read data into a pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Close database connection
conn.close()

# Print the DataFrame
print(df.head())
```

Ví dụ 3: Tính logP sử dụng RDKit (Calculate logP using RDKit)

SQL: (Không cần SQL ở đây, chúng ta sẽ sử dụng Python để đọc SMILES và tính logP)

Python:

```python
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import psycopg2

# Database connection details
db_host = "192.168.206.136"
db_name = "chembl_35"
db_user = "rd"
db_pass = "rd"
db_port = "5432"
```

```python
# Establish database connection
conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass, port=db_port)

# SQL query
sql_query = "SELECT chembl_id, canonical_smiles FROM molecule_dictionary LIMIT 100;"

# Read data into a pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Close database connection
conn.close()

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

# Apply logP calculation
df['logp'] = df['canonical_smiles'].apply(calculate_logp)

# Print the DataFrame with logP values
print(df.head())
```

Ví dụ 4: Tìm các hợp chất hoạt động trên một mục tiêu cụ thể (Find Compounds Active Against a Specific Target)

SQL:

```sql
-- Get compounds with IC50 values less than 1000 nM for target CHEMBL205
-- Lấy các hợp chất có giá trị IC50 nhỏ hơn 1000 nM cho mục tiêu CHEMBL205
SELECT md.chembl_id, act.standard_value
FROM molecule_dictionary md
JOIN activities act ON md.molregno = act.molregno
JOIN target_dictionary td ON act.tid = td.tid
WHERE td.target_chembl_id = 'CHEMBL205'
AND act.standard_type = 'IC50'
AND act.standard_units = 'nM'
AND act.standard_value <= 1000
AND act.standard_value::text ~ '^[0-9\.]+$'
LIMIT 100;
```

Python: (Bạn có thể đọc kết quả SQL vào DataFrame và thực hiện các phân tích tiếp theo)

```python
import pandas as pd
import psycopg2

# Database connection details
db_host = "192.168.206.136"
db_name = "chembl_35"
db_user = "rd"
db_pass = "rd"
db_port = "5432"

# Establish database connection
conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass, port=db_port)
```

```
# SQL query
sql_query = """
SELECT md.chembl_id, act.standard_value
FROM molecule_dictionary md
JOIN activities act ON md.molregno = act.molregno
JOIN target_dictionary td ON act.tid = td.tid
WHERE td.target_chembl_id = 'CHEMBL205'
AND act.standard_type = 'IC50'
AND act.standard_units = 'nM'
AND act.standard_value <= 1000
AND act.standard_value::text ~ '^[0-9\.]+$'
LIMIT 100;
"""

# Read data into a pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Close database connection
conn.close()

# Print the DataFrame
print(df.head())
```

Ví dụ 5: Xây dựng mô hình hồi quy đơn giản (Build a Simple Regression Model)

SQL: (Chúng ta sẽ sử dụng Python để chuẩn bị dữ liệu và xây dựng mô hình)

Python:

```
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, r2_score
import numpy as np
import psycopg2

# Database connection details
db_host = "192.168.206.136"
db_name = "chembl_35"
db_user = "rd"
db_pass = "rd"
db_port = "5432"

# Establish database connection
conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass, port=db_port)

# SQL query to fetch compound SMILES and IC50 values
sql_query = """
SELECT md.canonical_smiles, act.standard_value
FROM molecule_dictionary md
JOIN activities act ON md.molregno = act.molregno
JOIN target_dictionary td ON act.tid = td.tid
WHERE td.target_chembl_id = 'CHEMBL205'
AND act.standard_type = 'IC50'
AND act.standard_units = 'nM'
AND act.standard_value IS NOT NULL
AND act.standard_value::text ~ '^[0-9\.]+$'
```

```
LIMIT 100;
"""

# Read data into a pandas DataFrame
df = pd.read_sql_query(sql_query, conn)

# Close database connection
conn.close()


# Data Cleaning and Preprocessing
df.dropna(subset=['standard_value'], inplace=True)  # Remove rows with missing IC50
values
df = df[pd.to_numeric(df['standard_value'], errors='coerce').notna()] #remove non
numeric values
df['standard_value'] = pd.to_numeric(df['standard_value']) # Convert IC50 to numeric

# Function to calculate molecular descriptors using RDKit
def calculate_descriptors(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None
        descriptors = {desc_name: desc_func(mol) for desc_name, desc_func in
Descriptors.descList}
        return descriptors
    except:
        return None

# Apply descriptor calculation to each molecule
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Handle cases where descriptor calculation failed
df = df.dropna(subset=['descriptors'])

# Convert descriptors to a DataFrame and merge with the main DataFrame
descriptors_df = pd.DataFrame(df['descriptors'].tolist())
df = pd.concat([df, descriptors_df], axis=1)

# Prepare data for machine learning
X = df.drop(['canonical_smiles', 'standard_value', 'descriptors'], axis=1,
errors='ignore') #drop canonical_smiles
y = df['standard_value']

# Handle missing values (imputation)
X = X.replace([np.inf, -np.inf], np.nan) # Replace infinite values with NaN
X = X.fillna(X.mean())  # Replace NaN with the mean of each column

# 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)

# Train a linear regression 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)
r2 = r2_score(y_test, y_pred)

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

Quan trọng:

- Hãy nhớ thay đổi `db_host`, `db_name`, `db_user`, `db_pass` và `db_port` thành thông tin đăng nhập cơ sở dữ liệu thực tế của bạn.
- Bạn nên lưu trữ thông tin đăng nhập cơ sở dữ liệu một cách an toàn và không mã hóa chúng trực tiếp trong mã của bạn. Consider using environment variables or a configuration file.
- Các ví dụ này chỉ là điểm khởi đầu. Bạn có thể tùy chỉnh chúng để phù hợp với nhu cầu cụ thể của dự án của bạn.

Tôi hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án phân tích dữ liệu ChEMBL 35 của mình. Chúc bạn thành công!