PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Ánh xạ protein giữa người và chuột

Tuyệt vời! Với vai trò là một chuyên gia nghiên cứu và phát triển dược học, tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Dưới đây là phân tích, hướng dẫn song ngữ, code mẫu SQL và Python để giúp bạn bắt đầu, đồng thời giải quyết các vấn đề bạn đã đề cập.

Chủ đề: Topic_CheMBL_35_28

Tên chủ đề: Phân tích mối tương quan giữa các đặc tính lý hóa (Physicochemical Properties) và hoạt tính sinh học (Biological Activity) của các hợp chất trong ChEMBL 35.

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

Mục tiêu của chủ đề này là tìm hiểu xem các đặc tính lý hóa của một phân tử (ví dụ: trọng lượng phân tử, logP, số lượng liên kết hydro, v.v.) ảnh hưởng như thế nào đến hoạt tính sinh học của nó (ví dụ: IC50, Ki). Mô hình phân tích sẽ bao gồm các bước sau:

- **Thu thập và tiền xử lý dữ liệu:** Lấy dữ liệu từ cơ sở dữ liệu ChEMBL 35, làm sạch và chuyển đổi dữ liệu về hoạt tính sinh học (ví dụ: chuyển đổi IC50 sang pIC50).
- **Tính toán các đặc tính lý hóa:** Sử dụng RDKit để tính toán các đặc tính lý hóa cho mỗi phân tử.
- **Phân tích tương quan:** Sử dụng các phương pháp thống kê (ví dụ: hồi quy tuyến tính, hồi quy đa biến) để đánh giá mối tương quan giữa các đặc tính lý hóa và hoạt tính sinh học.
- **Xây dựng mô hình dự đoán:** (Tùy chọn) Xây dựng mô hình máy học để dự đoán hoạt tính sinh học dựa trên các đặc tính lý hóa.
- **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).

In English:

The goal of this topic is to understand how the physicochemical properties of a molecule (e.g., molecular weight, logP, number of hydrogen bonds, etc.) affect its biological activity (e.g., IC50, Ki). The analytical model will include the following steps:

- **Data Collection and Preprocessing:** Retrieve data from the ChEMBL 35 database, clean, and transform the biological activity data (e.g., convert IC50 to pIC50).
- **Calculation of Physicochemical Properties:** Use RDKit to calculate physicochemical properties for each molecule.
- **Correlation Analysis:** Use statistical methods (e.g., linear regression, multivariate regression) to assess the correlation between physicochemical properties and biological activity.
- **Building a Predictive Model:** (Optional) Build a machine learning model to predict biological activity based on physicochemical properties.
- **Model Evaluation:** Evaluate the performance of the model using appropriate metrics (e.g., R-squared, RMSE).

2. Hướng dẫn song ngữ (Bilingual Instructions):

Bước 1: Kết nối đến cơ sở dữ liệu ChEMBL 35 (Connect to ChEMBL 35 Database):

Sử dụng thông tin bạn cung cấp để kết nối đến cơ sở dữ liệu PostgreSQL.

```
import psycopg2

conn = psycopg2.connect(
    host="192.168.206.136",
    user="rd",
    password="rd",
    database="chembl_35"
)

cursor = conn.cursor()
```

In English:

Use the information you provided to connect to the PostgreSQL database.

Bước 2: Truy vấn dữ liệu từ ChEMBL 35 (Query Data from ChEMBL 35):

Sử dụng SQL để truy vấn dữ liệu cần thiết. **Lưu ý lỗi liên quan đến toán tử ~**. Toán tử ~ trong PostgreSQL là toán tử so sánh pattern matching (tương tự regex). Để khắc phục lỗi "ERROR: operator does not exist: numeric ~ unknown", bạn cần đảm bảo rằng cột `act.standard_value` phải là kiểu dữ liệu text trước khi sử dụng toán tử ~.

```
-- Lấy 100 dòng dữ liệu
SELECT
    cmp.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.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\.]+$' -- Ép kiểu về text trước khi so sánh
LIMIT 100;
```

In English:

Use SQL to query the necessary data. **Note the error related to the ~ operator.** The ~ operator in PostgreSQL is a pattern matching operator (similar to regex). To fix the error "ERROR: operator does not exist: numeric ~ unknown", you need to ensure that the `act.standard_value` column is of text data type before using the ~ operator. The `::text` cast ensures this.

Bước 3: Lưu dữ liệu vào file CSV (Save Data to CSV File):

Sử dụng `psycopg2` để lấy dữ liệu và lưu vào file CSV.

```
import csv
import os

base_path = "../data"  # Đảm bảo đường dẫn này đúng với cấu trúc thư mục của bạn

sql_query = """
SELECT
    cmp.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    compound_structures cmp
```

```
JOIN
    activities act ON cmp.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\.]+$'
LIMIT 100;
"""

cursor.execute(sql_query)
results = cursor.fetchall()

csv_file_path = os.path.join(base_path, "chembl_data.csv")

with open(csv_file_path, 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)
    csv_writer.writerow([i[0] for i in cursor.description])  # Ghi header
    csv_writer.writerows(results)

print(f"Dữ liệu đã được lưu vào: {csv_file_path}")

conn.close()
```

In English:

Use psycopg2 to retrieve the data and save it to a CSV file.

Bước 4: Đọc dữ liệu từ file CSV và tính toán đặc tính lý hóa bằng RDKit (Read Data from CSV File and Calculate Physicochemical Properties using RDKit):

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

# Đọc dữ liệu từ file CSV
csv_file_path = os.path.join(base_path, "chembl_data.csv")
df = pd.read_csv(csv_file_path)

# Hàm tính toán đặc tính lý hóa
def calculate_physicochemical_properties(chembl_id):
    try:
        # Lấy SMILES từ ChEMBL (cần một truy vấn SQL khác)
        sql_query = f"""
        SELECT cs.canonical_smiles
        FROM compound_structures cs
        WHERE cs.chembl_id = '{chembl_id}'
        """
        cursor = conn.cursor()
        cursor.execute(sql_query)
        smiles = cursor.fetchone()[0]

        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None

        # Tính toán các đặc tính lý hóa
        mw = Descriptors.MolWt(mol)
        logp = Chem.Crippen.MolLogP(mol)
        hbd = Chem.rdMolDescriptors.CalcNumHBD(mol)
```

```
        hba = Chem.rdMolDescriptors.CalcNumHBA(mol)
        return mw, logp, hbd, hba
    except Exception as e:
        print(f"Lỗi khi tính toán đặc tính cho {chembl_id}: {e}")
        return None

# Áp dụng hàm tính toán đặc tính lý hóa cho mỗi hợp chất
df[['molecular_weight', 'logp', 'hbd', 'hba']] = df['chembl_id'].apply(lambda x:
pd.Series(calculate_physicochemical_properties(x)))

# Chuyển đổi standard_value sang pIC50
df = df.dropna(subset=['standard_value'])
df['standard_value'] = pd.to_numeric(df['standard_value'])
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Chuyển đổi nM sang M

# In ra DataFrame
print(df.head())
```

In English:

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

# Read data from CSV file
csv_file_path = os.path.join(base_path, "chembl_data.csv")
df = pd.read_csv(csv_file_path)

# Function to calculate physicochemical properties
def calculate_physicochemical_properties(chembl_id):
    try:
        # Get SMILES from ChEMBL (requires another SQL query)
        sql_query = f"""
        SELECT cs.canonical_smiles
        FROM compound_structures cs
        WHERE cs.chembl_id = '{chembl_id}'
        """
        cursor = conn.cursor()
        cursor.execute(sql_query)
        smiles = cursor.fetchone()[0]

        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None

        # Calculate physicochemical properties
        mw = Descriptors.MolWt(mol)
        logp = Chem.Crippen.MolLogP(mol)
        hbd = Chem.rdMolDescriptors.CalcNumHBD(mol)
        hba = Chem.rdMolDescriptors.CalcNumHBA(mol)
        return mw, logp, hbd, hba
    except Exception as e:
        print(f"Error calculating properties for {chembl_id}: {e}")
        return None

# Apply the function to each compound
df[['molecular_weight', 'logp', 'hbd', 'hba']] = df['chembl_id'].apply(lambda x:
pd.Series(calculate_physicochemical_properties(x)))
```

```
# Convert standard_value to pIC50
df = df.dropna(subset=['standard_value'])
df['standard_value'] = pd.to_numeric(df['standard_value'])
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Convert nM to M

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

Bước 5: Phân tích tương quan (Correlation Analysis):

```
# Tính toán ma trận tương quan
correlation_matrix = df[['pIC50', 'molecular_weight', 'logp', 'hbd', 'hba']].corr()
print(correlation_matrix)

import matplotlib.pyplot as plt
import seaborn as sns

sns.heatmap(correlation_matrix, annot=True)
plt.show()
```

In English:

```
# Calculate the correlation matrix
correlation_matrix = df[['pIC50', 'molecular_weight', 'logp', 'hbd', 'hba']].corr()
print(correlation_matrix)

import matplotlib.pyplot as plt
import seaborn as sns

sns.heatmap(correlation_matrix, annot=True)
plt.show()
```

Bước 6: Xây dựng mô hình hồi quy (Regression Model):

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Chuẩn bị dữ liệu
X = df[['molecular_weight', 'logp', 'hbd', 'hba']].fillna(0)  # Điền giá trị NaN bằng
0
y = df['pIC50'].fillna(0)

# 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)

# Xây dựng mô hình hồi quy tuyến tính
model = LinearRegression()
model.fit(X_train, y_train)

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

# Đánh giá mô hình
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}")
```

In English:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Prepare the data
X = df[['molecular_weight', 'logp', 'hbd', 'hba']].fillna(0)  # Fill NaN values with 0
y = df['pIC50'].fillna(0)

# Split the 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)

# Build a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the test set
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 về lỗi sklearn: Nếu bạn gặp lỗi liên quan đến `squared=False` trong `mean_squared_error`, hãy đảm bảo rằng bạn đang sử dụng phiên bản scikit-learn mới nhất. Nếu không, bạn có thể tính căn bậc hai của MSE để có RMSE.

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

Dưới đây là 5 ví dụ về code SQL và Python mẫu:

Ví dụ 1: Lấy số lượng hợp chất cho mỗi loại hoạt tính (SQL):

```
SELECT standard_type, COUNT(*) FROM activities GROUP BY standard_type;
```

In English:

```
SELECT standard_type, COUNT(*) FROM activities GROUP BY standard_type;
```

Ví dụ 1: Lấy SMILES canonical và molecular weight của các chất (SQL, Python):

```
SELECT
    cmp.chembl_id,
    cs.canonical_smiles
FROM
    compound_structures cs join compound_properties cmp
    ON cs.molregno = cmp.molregno
LIMIT 10;
```

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

def get_molecular_weight(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
```

```
    return None

sql_query = """
SELECT
    cmp.chembl_id,
    cs.canonical_smiles
FROM
    compound_structures cs join compound_properties cmp
    ON cs.molregno = cmp.molregno
LIMIT 10;
"""

cursor.execute(sql_query)
results = cursor.fetchall()

df = pd.DataFrame(results, columns=['chembl_id', 'smiles'])
df['molecular_weight'] = df['smiles'].apply(get_molecular_weight)

print(df)
```

Ví dụ 2: Tính toán LogP cho một danh sách các chất (Python):

```
from rdkit import Chem
from rdkit.Chem import Crippen

smiles_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']

for smiles in smiles_list:
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        logp = Crippen.MolLogP(mol)
        print(f"LogP for {smiles}: {logp}")
    else:
        print(f"Invalid SMILES: {smiles}")
```

In English:

```
from rdkit import Chem
from rdkit.Chem import Crippen

smiles_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']

for smiles in smiles_list:
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        logp = Crippen.MolLogP(mol)
        print(f"LogP for {smiles}: {logp}")
    else:
        print(f"Invalid SMILES: {smiles}")
```

Ví dụ 3: Lọc các chất có trọng lượng phân tử nằm trong một khoảng nhất định (SQL, Python):

```
-- Cần tạo một view hoặc bảng tạm để lưu trữ trọng lượng phân tử
-- (Ví dụ này chỉ minh họa logic, cần điều chỉnh cho phù hợp với cấu trúc DB)
SELECT chembl_id FROM compound_properties WHERE molecular_weight > 200 AND
molecular_weight < 500;

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

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

sql_query = """
SELECT
    cmp.chembl_id,
    cs.canonical_smiles
FROM
    compound_structures cs join compound_properties cmp
    ON cs.molregno = cmp.molregno
LIMIT 100;
"""

cursor.execute(sql_query)
results = cursor.fetchall()

df = pd.DataFrame(results, columns=['chembl_id', 'smiles'])
df['molecular_weight'] = df['smiles'].apply(calculate_mw)

filtered_df = df[(df['molecular_weight'] > 200) & (df['molecular_weight'] < 500)]
print(filtered_df)
```

Ví dụ 4: Tính toán số lượng liên kết hydro cho và nhận (Python):

```
from rdkit import Chem
from rdkit.Chem import rdMolDescriptors

smiles = 'CC(=O)Oc1ccccc1C(=O)O'
mol = Chem.MolFromSmiles(smiles)

if mol:
    hbd = rdMolDescriptors.CalcNumHBD(mol)
    hba = rdMolDescriptors.CalcNumHBA(mol)
    print(f"Hydrogen Bond Donors: {hbd}")
    print(f"Hydrogen Bond Acceptors: {hba}")
else:
    print("Invalid SMILES")
```

In English:

```
from rdkit import Chem
from rdkit.Chem import rdMolDescriptors

smiles = 'CC(=O)Oc1ccccc1C(=O)O'
mol = Chem.MolFromSmiles(smiles)

if mol:
    hbd = rdMolDescriptors.CalcNumHBD(mol)
    hba = rdMolDescriptors.CalcNumHBA(mol)
    print(f"Hydrogen Bond Donors: {hbd}")
    print(f"Hydrogen Bond Acceptors: {hba}")
else:
    print("Invalid SMILES")
```

Ví dụ 5: Phân tích tương quan giữa LogP và hoạt tính (SQL, Python):

(SQL - Ý tưởng, cần điều chỉnh theo schema):

```
-- Ví dụ này cần kết hợp dữ liệu từ nhiều bảng và có thể yêu cầu tạo view
-- Mục đích là minh họa ý tưởng, cần điều chỉnh cho phù hợp với database schema
-- SELECT corr(logp, pIC50) FROM ... JOIN ... WHERE ...;
```

```
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Crippen

# Giả sử bạn đã có DataFrame 'df' chứa SMILES và pIC50
# Ví dụ:
data = {'smiles': ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO'],
        'pIC50': [6.5, 7.0, 5.5]}
df = pd.DataFrame(data)

# Tính LogP
df['logp'] = df['smiles'].apply(lambda x: Crippen.MolLogP(Chem.MolFromSmiles(x)) if
Chem.MolFromSmiles(x) else None)

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

# Tính toán tương quan
correlation = df['logp'].corr(df['pIC50'])
print(f"Correlation between LogP and pIC50: {correlation}")
```

Lưu ý quan trọng:

- Thay thế các đường dẫn file và thông tin kết nối cơ sở dữ liệu bằng thông tin chính xác của bạn.
- Các truy vấn SQL cần được điều chỉnh để phù hợp với cấu trúc cơ sở dữ liệu ChEMBL 35.
- Bạn có thể cần cài đặt các thư viện Python cần thiết (ví dụ: `psycopg2`, `rdkit`, `pandas`, `scikit-learn`) bằng pip.
- Hãy nhớ đóng kết nối cơ sở dữ liệu sau khi hoàn thành.
- Để sử dụng lại các đối tượng cursor , hãy tạo lại đối tượng mới sau khi close.
- Với lượng dữ liệu lớn, hãy xem xét sử dụng các kỹ thuật tối ưu hóa hiệu suất.

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