PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

Copyright 2025 | Nghiên Cứu Thuốc | www.nghiencuuthuoc.com | Zalo: +84888999311 |

🧩 Topic: Biểu đồ violin cho phổ hoạt tính đa đích

Tuyệt vời! Tôi rất vui được tham gia dự án thú vị này với vai trò Chuyên gia R&D Dược học. Phân tích dữ liệu từ ChEMBL bằng RDKit là một hướng đi rất tiềm năng để khám phá và phát triển thuốc mới.

Dưới đây là phân tích, hướng dẫn song ngữ, code mẫu SQL và Python, cùng với các ví dụ để bạn bắt đầu dự án "Topic_CheMBL_35_78" một cách hiệu quả.

1. Phân tích mô hình (Analysis Model):

Mục tiêu của dự án này là tận dụng dữ liệu ChEMBL 35 và các công cụ tin sinh học như RDKit để:

- **Tìm kiếm các hợp chất có hoạt tính sinh học tiềm năng:** Xác định các hợp chất có khả năng tương tác với mục tiêu (target) mong muốn.
- **Xây dựng mô hình QSAR/QSPR:** Phát triển các mô hình định lượng mối liên hệ giữa cấu trúc và hoạt tính/tính chất của hợp chất.
- **Dự đoán hoạt tính/tính chất của hợp chất mới:** Sử dụng mô hình đã xây dựng để dự đoán các đặc tính của các hợp chất chưa được thử nghiệm.
- **Hỗ trợ thiết kế thuốc:** Đưa ra các gợi ý về cấu trúc để tối ưu hóa hoạt tính và tính chất của thuốc.

Mô hình phân tích bao gồm các bước chính:

1. **Thu thập và chuẩn bị dữ liệu:**
    - Lấy dữ liệu từ cơ sở dữ liệu ChEMBL (thông qua SQL).
    - Làm sạch và tiền xử lý dữ liệu (loại bỏ dữ liệu trùng lặp, xử lý giá trị thiếu, chuẩn hóa dữ liệu).
2. **Tính toán đặc trưng phân tử (Molecular Feature Calculation):**
    - Sử dụng RDKit để tính toán các đặc trưng phân tử (descriptors) từ cấu trúc hóa học của các hợp chất (ví dụ: trọng lượng phân tử, logP, số lượng liên kết, diện tích bề mặt, v.v.).
3. **Phân tích khám phá dữ liệu (Exploratory Data Analysis - EDA):**
    - Thống kê mô tả (descriptive statistics)
    - Trực quan hóa dữ liệu (data visualization)
    - Phân tích tương quan (correlation analysis)
4. **Xây dựng mô hình (Model Building):**
    - Lựa chọn thuật toán học máy phù hợp (ví dụ: hồi quy tuyến tính, Support Vector Machines, Random Forest, v.v.).
    - Chia dữ liệu thành tập huấn luyện (training set) và tập kiểm tra (test set).
    - Huấn luyện mô hình trên tập huấn luyện.
    - Tối ưu hóa mô hình (ví dụ: điều chỉnh siêu tham số).
5. **Đánh giá mô hình (Model Evaluation):**
    - Sử dụng tập kiểm tra để đánh giá hiệu năng của mô hình (ví dụ: R-squared, RMSE, MAE, v.v.).
    - So sánh hiệu năng của các mô hình khác nhau.
6. **Ứng dụng mô hình (Model Application):**
    - Sử dụng mô hình để dự đoán hoạt tính/tính chất của các hợp chất mới.
    - Phân tích các hợp chất có hoạt tính cao để tìm ra các đặc trưng quan trọng.

- Đề xuất các cấu trúc hóa học mới có tiềm năng trở thành thuốc.

2. Hướng dẫn song ngữ (Bilingual Guide):

2.1. Chuẩn bị dữ liệu (Data Preparation):

- **SQL:** Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL. Ví dụ: `sql -- Lấy thông tin về các hợp chất và hoạt tính của chúng đối với một mục tiêu cụ thể -- Get information about compounds and their activities against a specific target SELECT md.chembl_id, cs.canonical_smiles, act.standard_type, 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' -- Lọc theo loại hoạt tính (ví dụ: IC50) / Filter by activity type (e.g., IC50) AND act.standard_units = 'nM' -- Lọc theo đơn vị (ví dụ: nM) / Filter by units (e.g., nM) LIMIT 100;`

- **Python:** Sử dụng thư viện `psycopg2` để kết nối đến cơ sở dữ liệu PostgreSQL và thực thi các truy vấn SQL. Sử dụng `pandas` để xử lý dữ liệu trả về.

```python
# Kết nối đến cơ sở dữ liệu PostgreSQL
# Connect to the PostgreSQL database
import psycopg2
import pandas as pd

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

# Tạo một con trỏ (cursor) để thực thi các truy vấn SQL
# Create a cursor to execute SQL queries
cur = conn.cursor()

# Thực thi truy vấn SQL
# Execute the SQL query
sql_query = """
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    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'
LIMIT 100;
"""
cur.execute(sql_query)

# Lấy kết quả truy vấn
# Fetch the query results
```

```
results = cur.fetchall()

# Chuyển đổi kết quả thành DataFrame của pandas
# Convert the results to a pandas DataFrame
df = pd.DataFrame(results, columns=['chembl_id', 'canonical_smiles',
'standard_type', 'standard_value', 'standard_units'])

# In ra DataFrame
# Print the DataFrame
print(df.head())

# Đóng kết nối
# Close the connection
cur.close()
conn.close()
```

2.2. Tính toán đặc trưng phân tử (Molecular Feature Calculation):

- **Python:** Sử dụng RDKit để tính toán các đặc trưng phân tử từ SMILES.

```
# Tính toán các đặc trưng phân tử bằng RDKit
# Calculate molecular features using RDKit
from rdkit import Chem
from rdkit.Chem import Descriptors

def calculate_descriptors(smiles):
    """
    Tính toán các đặc trưng phân tử từ SMILES.
    Calculates molecular descriptors from SMILES.
    """
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    descriptors['mol_weight'] = Descriptors.MolWt(mol)
    descriptors['logp'] = Descriptors.MolLogP(mol)
    # Thêm các đặc trưng khác tùy ý / Add other descriptors as needed
    return descriptors

# Áp dụng hàm tính toán đặc trưng cho mỗi SMILES trong DataFrame
# Apply the descriptor calculation function to each SMILES in the DataFrame
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Tạo các cột riêng biệt cho từng đặc trưng
# Create separate columns for each descriptor
df = pd.concat([df, df['descriptors'].apply(pd.Series)], axis=1)

print(df.head())
```

2.3. Xây dựng mô hình (Model Building):

- **Python:** Sử dụng scikit-learn để xây dựng mô hình học máy.

```
# Xây dựng mô hình học máy
# Build a machine learning model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Loại bỏ các hàng có giá trị NaN
```

```
# Remove rows with NaN values
df = df.dropna()

# Chuẩn bị dữ liệu cho mô hình
# Prepare data for the model
X = df[['mol_weight', 'logp']]  # Chọn các đặc trưng / Select features
y = df['standard_value'].astype(float)  # Chọn biến mục tiêu / Select target
variable

# 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(X, y, test_size=0.2,
random_state=42)

# Khởi tạo và huấn luyện mô hình
# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
# Predict 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) # nếu scikit-learn < 0.20, bỏ
squared=False
print(f"Mean Squared Error: {mse}")
```

Giải thích lỗi và cách sửa:

- **Lỗi a:** **`ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`**
  - **Nguyên nhân:** Lỗi này xảy ra do bạn đang sử dụng toán tử ~ (regex match) trên một cột kiểu số (`numeric`). PostgreSQL không tự động chuyển đổi kiểu dữ liệu trong trường hợp này.
  - **Cách sửa:** Chuyển đổi cột `standard_value` sang kiểu text trước khi so sánh bằng regex.

```
AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'
```

- **Lỗi b:** **`phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean_squared_error`**
  - **Nguyên nhân:** Tham số `squared=False` chỉ có trong các phiên bản scikit-learn mới hơn (từ 0.20 trở lên).
  - **Cách sửa:** Nếu bạn đang sử dụng phiên bản scikit-learn cũ, hãy bỏ tham số `squared=False` và lấy căn bậc hai của kết quả để có RMSE (Root Mean Squared Error).

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

Hoặc, nâng cấp scikit-learn lên phiên bản mới nhất nếu có thể.

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

Dưới đây là một ví dụ hoàn chỉnh hơn, bao gồm cả tiền xử lý dữ liệu và lưu kết quả:

SQL:

```
-- Lấy dữ liệu và lọc các giá trị hợp lệ
-- Get data and filter for valid values
```

```sql
DROP TABLE IF EXISTS tmp_chembl_data;
CREATE TEMP TABLE tmp_chembl_data AS
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    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 CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$' -- Lọc các giá trị số / Filter
for numeric values
LIMIT 100;

-- Xuất dữ liệu ra file CSV
-- Export data to CSV file
\copy (SELECT * FROM tmp_chembl_data) TO '../data/chembl_ic50_data.csv' WITH CSV
HEADER;
```

Python:

```python
# Import thư viện
# Import libraries
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
import os

# Định nghĩa đường dẫn cơ sở
# Define base path
base_path = "." # Thay đổi nếu cần / Change if needed

# Đọc dữ liệu từ file CSV
# Read data from CSV file
data_path = os.path.join(base_path, "../data/chembl_ic50_data.csv")
df = pd.read_csv(data_path)

# Hàm tính toán descriptors
# Descriptor calculation function
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    try:
        descriptors['mol_weight'] = Descriptors.MolWt(mol)
        descriptors['logp'] = Descriptors.MolLogP(mol)
        descriptors['hbd'] = Descriptors.NumHDonors(mol)
        descriptors['hba'] = Descriptors.NumHAcceptors(mol)
```

```
    except:
        return None
    return descriptors

# Áp dụng hàm tính toán descriptors
# Apply descriptor calculation function
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)
df = pd.concat([df, df['descriptors'].apply(pd.Series)], axis=1)

# Loại bỏ các hàng có giá trị NaN
# Remove rows with NaN values
df = df.dropna()

# Chuẩn bị dữ liệu cho mô hình
# Prepare data for the model
X = df[['mol_weight', 'logp', 'hbd', 'hba']]
y = df['standard_value'].astype(float)

# 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(X, y, test_size=0.2,
random_state=42)

# Xây dựng mô hình
# Build the model
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán và đánh giá mô hình
# Predict and evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

4. Ví dụ (Examples):

Dưới đây là 5 ví dụ về các truy vấn SQL và đoạn code Python mà bạn có thể sử dụng để khám phá dữ liệu ChEMBL và xây dựng mô hình:

Ví dụ 1: Lọc theo mục tiêu (Target-based filtering)

- **SQL:** `sql -- Lấy các hợp chất có hoạt tính chống lại một mục tiêu cụ thể (ví dụ: EGFR) -- Get compounds with activity against a specific target (e.g., EGFR) SELECT md.chembl_id, cs.canonical_smiles, act.standard_type, act.standard_value FROM molecule_dictionary md JOIN compound_structures cs ON md.molregno = cs.molregno JOIN activities act ON md.molregno = act.molregno JOIN target_dictionary td ON act.tid = td.tid WHERE td.chembl_id = 'CHEMBL203' -- EGFR AND act.standard_type = 'IC50' AND act.standard_units = 'nM' LIMIT 100;`
- **Python:** (Sử dụng dữ liệu đã lấy từ SQL) `python # Lọc DataFrame để chỉ giữ lại các hợp chất có IC50 < 100 nM # Filter the DataFrame to only keep compounds with IC50 < 100 nM df_filtered = df[df['standard_value'].astype(float) < 100] print(df_filtered.head())`

Ví dụ 2: Tính toán số lượng vòng (Calculating Ring Count)

- **Python:**

```
from rdkit.Chem import rdMolDescriptors

def calculate_ring_count(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    return rdMolDescriptors.CalcNumRings(mol)

df['ring_count'] = df['canonical_smiles'].apply(calculate_ring_count)
print(df.head())
```

Ví dụ 3: Phân tích tương quan (Correlation Analysis)

- **Python:**

```
import seaborn as sns
import matplotlib.pyplot as plt

# Tính toán ma trận tương quan
# Calculate the correlation matrix
corr_matrix = df[['standard_value', 'mol_weight', 'logp', 'hbd', 'hba',
'ring_count']].corr()

# Vẽ ma trận tương quan
# Plot the correlation matrix
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.show()
```

Ví dụ 4: Mô hình hóa bằng Random Forest (Random Forest Modeling)

- **Python:**

```
from sklearn.ensemble import RandomForestRegressor

# Khởi tạo và huấn luyện mô hình Random Forest
# Initialize and train the Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Dự đoán và đánh giá mô hình
# Predict and evaluate the model
y_pred = rf_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Random Forest Mean Squared Error: {mse}")
```

Ví dụ 5: Lưu mô hình (Saving the Model)

- **Python:**

```
import pickle

# Lưu mô hình đã huấn luyện
# Save the trained model
model_filename = os.path.join(base_path, "linear_regression_model.pkl")
pickle.dump(model, open(model_filename, 'wb'))

# Load mô hình đã lưu (ví dụ)
# Load the saved model (example)
loaded_model = pickle.load(open(model_filename, 'rb'))
```

Lưu ý quan trọng:

- **Số lượng dữ liệu:** 100 dòng dữ liệu là rất ít để xây dựng một mô hình đáng tin cậy. Bạn nên cố gắng lấy nhiều dữ liệu hơn nếu có thể (sau khi đã thử nghiệm thành công với bộ dữ liệu nhỏ).
- **Tiền xử lý dữ liệu:** Dữ liệu ChEMBL có thể chứa nhiều giá trị ngoại lệ (outliers) và sai sót. Hãy dành thời gian để làm sạch và tiền xử lý dữ liệu một cách cẩn thận.
- **Lựa chọn đặc trưng:** Việc lựa chọn các đặc trưng phân tử phù hợp là rất quan trọng để xây dựng một mô hình tốt. Hãy thử nghiệm với nhiều loại đặc trưng khác nhau và sử dụng các phương pháp lựa chọn đặc trưng (feature selection) để tìm ra tập hợp các đặc trưng tốt nhất.
- **Đánh giá mô hình:** Sử dụng các phương pháp đánh giá mô hình phù hợp (ví dụ: cross-validation) để đảm bảo rằng mô hình của bạn có khả năng khái quát hóa tốt.
- **Tài liệu RDKit và scikit-learn:** Hãy tham khảo tài liệu chính thức của RDKit và scikit-learn để tìm hiểu thêm về các tính năng và tùy chọn của các thư viện này.

Chúc bạn thành công với dự án của mình! Nếu bạn có bất kỳ câu hỏi nào, đừng ngần ngại hỏi. Tôi luôn sẵn lòng giúp đỡ.