PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Ánh xạ thuốc ChEMBL sang mã ATC của WHO

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 R&D dược học, tôi hiểu tầm quan trọng của việc phân tích dữ liệu một cách hiệu quả để thúc đẩy quá trình phát triển thuốc.

Dưới đây là phân tích chi tiết, hướng dẫn song ngữ và các ví dụ code để bạn có thể bắt đầu:

1. Phân tích Mô hình (Analysis Model)

Chúng ta sẽ tập trung vào việc sử dụng dữ liệu ChEMBL 35 kết hợp với RDKit để tạo ra các mô hình phân tích sau:

- **Phân tích Hoạt tính (Activity Analysis):** Xác định các hợp chất có hoạt tính sinh học cao đối với một mục tiêu cụ thể.
- **Phân tích SAR (Structure-Activity Relationship):** Tìm hiểu mối quan hệ giữa cấu trúc hóa học của hợp chất và hoạt tính sinh học của nó.
- **Dự đoán Hoạt tính (Activity Prediction):** Xây dựng mô hình dự đoán hoạt tính của các hợp chất mới dựa trên dữ liệu hiện có.
- **Phân tích Tương đồng (Similarity Analysis):** Tìm kiếm các hợp chất tương tự về cấu trúc hoặc hoạt tính.
- **Phân tích QSAR/QSPR (Quantitative Structure-Activity/Property Relationship):** Xây dựng mô hình định lượng mối quan hệ giữa cấu trúc và hoạt tính/tính chất.

2. Hướng dẫn Song ngữ (Bilingual Instructions)

Tổng quan:

- **Tiếng Việt:** Chúng ta sẽ sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35, sau đó sử dụng Python và RDKit để xử lý và phân tích dữ liệu.
- **English:** We will use SQL to query data from the ChEMBL 35 database, then use Python and RDKit to process and analyze the data.

Chi tiết:

1. **Tiếng Việt:** Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35, tập trung vào các bảng như `activities`, `molecule_dictionary`, và `compound_structures`.
    - **English:** Use SQL to query data from the ChEMBL 35 database, focusing on tables such as `activities`, `molecule_dictionary`, and `compound_structures`.
2. **Tiếng Việt:** Sử dụng Python và thư viện RDKit để xử lý dữ liệu hóa học, tính toán các đặc trưng phân tử (molecular descriptors), và tạo fingerprint.
    - **English:** Use Python and the RDKit library to process chemical data, calculate molecular descriptors, and generate fingerprints.
3. **Tiếng Việt:** Sử dụng các thư viện Python như scikit-learn để xây dựng các mô hình học máy, chẳng hạn như hồi quy tuyến tính, random forest, hoặc SVM, để dự đoán hoạt tính.
    - **English:** Use Python libraries like scikit-learn to build machine learning models, such as linear regression, random forest, or SVM, to predict activity.
4. **Tiếng Việt:** Trực quan hóa dữ liệu bằng các thư viện như matplotlib và seaborn để hiểu rõ hơn về dữ liệu và kết quả mô hình.

- **English:** Visualize data using libraries like matplotlib and seaborn to better understand the data and model results.

3. Code SQL và Python (SQL and Python Code)

a. SQL (Ví dụ)

```sql
-- Lấy 100 dòng dữ liệu hoạt tính của một mục tiêu cụ thể (ví dụ: EGFR)
-- Get 100 activity data points for a specific target (e.g., EGFR)
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.target_chembl_id = 'CHEMBL203' -- EGFR
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND act.standard_value::text ~ '^[0-9\.]+$' -- Giải quyết lỗi "operator does not
exist"
LIMIT 100;
```

b. Python (Ví dụ)

```python
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
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, r2_score
from sklearn.preprocessing import StandardScaler

# Đường dẫn cơ sở của dự án
base_path = "."  # Thay đổi nếu cần
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")

# Đọc dữ liệu từ file CSV
# Read data from CSV file
csv_file = os.path.join(data_path, "CHEMBL203_IC50_100.csv") # Thay đổi tên file nếu
cần
df = pd.read_csv(csv_file)

# Xử lý dữ liệu RDKit
# Process RDKit data
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        return Descriptors.MolWt(mol), Descriptors.MolLogP(mol)
```

```
    else:
        return None, None

df['mol_weight'], df['logP'] =
zip(*df['canonical_smiles'].apply(calculate_descriptors))
df = df.dropna()

# Chuẩn bị dữ liệu cho mô hình
# Prepare data for the model
X = df[['mol_weight', 'logP']]
y = np.log10(df['standard_value'].astype(float)) # Chuyển đổi về log scale

# 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)

# Chuẩn hóa dữ liệu
# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Xây dựng mô hình hồi quy tuyến tính
# Build a linear regression 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)
r2 = r2_score(y_test, y_pred)

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

4. Giải quyết Lỗi (Error Resolution)

a. Lỗi SQL: `operator does not exist: numeric ~ unknown`

Lỗi này xảy ra do PostgreSQL không thể so sánh kiểu dữ liệu `numeric` với kiểu `unknown` (trong trường hợp này, là kết quả của biểu thức chính quy). Để giải quyết, bạn cần ép kiểu dữ liệu `standard_value` về kiểu `text` trước khi so sánh:

```
AND act.standard_value::text ~ '^[0-9\.]+$'
```

b. Lỗi Python: `squared=False` trong `mean_squared_error`

Nếu bạn đang sử dụng phiên bản scikit-learn cũ, tham số `squared=False` không được hỗ trợ trong hàm `mean_squared_error`. Để giải quyết, bạn có thể tính căn bậc hai của MSE để có RMSE:

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

Hoặc, nâng cấp scikit-learn lên phiên bản mới nhất:

```
pip install -U scikit-learn
```

5. Ví dụ Code (Code Examples)

a. SQL (5 ví dụ)

1. **Lấy thông tin cơ bản của các hợp chất:**

```sql
-- Get basic information of compounds
SELECT chembl_id, pref_name FROM molecule_dictionary LIMIT 10;
```

2. **Lấy dữ liệu hoạt tính cho một hợp chất cụ thể:**

```sql
-- Get activity data for a specific compound
SELECT standard_type, standard_value, standard_units
FROM activities
WHERE molregno = (SELECT molregno FROM molecule_dictionary WHERE chembl_id =
'CHEMBL1')
LIMIT 10;
```

3. **Tìm các hợp chất có hoạt tính ức chế enzyme:**

```sql
-- Find compounds with enzyme inhibition activity
SELECT md.chembl_id, act.standard_value, act.standard_units
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50' AND act.target_chembl_id = 'CHEMBL203' LIMIT 10;
```

4. **Lấy cấu trúc SMILES của các hợp chất:**

```sql
-- Get SMILES structures of compounds
SELECT md.chembl_id, cs.canonical_smiles
FROM molecule_dictionary md
JOIN compound_structures cs ON md.molregno = cs.molregno LIMIT 10;
```

5. **Tìm các hợp chất có trọng lượng phân tử lớn hơn 500:**

```sql
-- Find compounds with molecular weight greater than 500
SELECT md.chembl_id, cs.canonical_smiles
FROM molecule_dictionary md
JOIN compound_structures cs ON md.molregno = cs.molregno
WHERE md.molregno IN (SELECT molregno FROM compound_properties WHERE mw_freebase >
500) LIMIT 10;
```

b. Python (5 ví dụ)

1. **Tính fingerprint của một phân tử:**

```python
# Calculate fingerprint of a molecule
from rdkit import Chem
from rdkit.Chem import AllChem

smiles = 'CC(=O)Oc1ccccc1C(=O)O'
mol = Chem.MolFromSmiles(smiles)
fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=1024)
print(fp.ToBitString())
```

2. **Tính toán các đặc trưng phân tử đơn giản:**

```python
# Calculate simple molecular descriptors
from rdkit import Chem
from rdkit.Chem import Descriptors

smiles = 'CC(=O)Oc1ccccc1C(=O)O'
mol = Chem.MolFromSmiles(smiles)
print(Descriptors.MolWt(mol))
print(Descriptors.MolLogP(mol))
```

3. **Tìm kiếm các phân tử tương tự:**

```python
# Find similar molecules
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit import DataStructs

smiles_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']
mols = [Chem.MolFromSmiles(s) for s in smiles_list]
fps = [AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=1024) for mol in mols]

query_smiles = 'c1ccccc1C(=O)O'
query_mol = Chem.MolFromSmiles(query_smiles)
query_fp = AllChem.GetMorganFingerprintAsBitVect(query_mol, 2, nBits=1024)

similarities = [DataStructs.TanimotoSimilarity(query_fp, fp) for fp in fps]
print(similarities)
```

4. **Sử dụng Pandas để đọc và xử lý dữ liệu:**

```python
# Use Pandas to read and process data
import pandas as pd

data = {'smiles': ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO'],
        'activity': [5.0, 6.0, 7.0]}
df = pd.DataFrame(data)
print(df.head())
```

5. **Trực quan hóa dữ liệu với Matplotlib:**

```python
# Visualize data with Matplotlib
import matplotlib.pyplot as plt
import pandas as pd

data = {'smiles': ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO'],
        'activity': [5.0, 6.0, 7.0]}
df = pd.DataFrame(data)

plt.bar(df['smiles'], df['activity'])
plt.xlabel('Smiles')
plt.ylabel('Activity')
plt.title('Activity vs Smiles')
plt.show()
```

Lưu ý Quan trọng:

- **Đường dẫn:** Hãy chắc chắn rằng bạn đã điều chỉnh đường dẫn `base_path`, `data_path`, và `notebook_path` cho phù hợp với cấu trúc thư mục dự án của bạn.
- **Tên file:** Thay đổi tên file CSV trong code Python để phù hợp với tên file bạn đã lưu từ kết quả truy vấn SQL.
- **Phiên bản thư viện:** Đảm bảo bạn đã cài đặt các phiên bản thư viện cần thiết (RDKit, scikit-learn, pandas, matplotlib, seaborn) và chúng tương thích với nhau.
- **Hiệu suất:** Với 100 dòng dữ liệu, bạn có thể thử nghiệm với nhiều mô hình khác nhau. Tuy nhiên, khi làm việc với dữ liệu lớn hơn, hãy cân nhắc sử dụng các kỹ thuật tối ưu hóa để cải thiện hiệu suất.

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