PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Chuyển SMILES từ ChEMBL thành InChIKey

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 rõ tầm quan trọng của việc khai thác dữ liệu ChEMBL để đẩy nhanh quá trình phát triển thuốc. Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu SQL và Python, cùng với các ví dụ cụ thể để bạn tham khảo.

Tổng quan dự án:

Chúng ta sẽ tập trung vào việc truy vấn, xử lý và phân tích dữ liệu từ cơ sở dữ liệu ChEMBL 35 sử dụng PostgreSQL và RDKit trong môi trường Jupyter Notebook. Mục tiêu là trích xuất thông tin có giá trị để hỗ trợ quá trình nghiên cứu và phát triển thuốc.

1. Phân tích mô hình (Analysis Model):

Chúng ta sẽ sử dụng mô hình phân tích dữ liệu cơ bản, 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 cơ sở dữ liệu ChEMBL.
- **Data Cleaning and Preprocessing (Làm sạch và tiền xử lý dữ liệu):** Sử dụng Python và RDKit để làm sạch dữ liệu, xử lý các giá trị thiếu, chuẩn hóa dữ liệu và tính toán các thuộc tính phân tử.
- **Exploratory Data Analysis (EDA) (Phân tích khám phá dữ liệu):** Sử dụng Python để khám phá dữ liệu, trực quan hóa các mối quan hệ giữa các biến và xác định các mẫu quan trọng.
- **Modeling (Mô hình hóa):** Xây dựng các mô hình dự đoán (ví dụ: mô hình hồi quy, mô hình phân loại) để dự đoán hoạt tính sinh học của các hợp chất dựa trên các thuộc tính phân tử của chúng.

2. Hướng dẫn song ngữ (Bilingual Guidance):

2.1. Thiết lập kết nối cơ sở dữ liệu (Setting up database connection):

- **English:** Use the `psycopg2` library in Python to connect to the PostgreSQL database.
- **Tiếng Việt:** Sử dụng thư viện `psycopg2` trong Python để kết nối đến cơ sở dữ liệu PostgreSQL.

2.2. Truy vấn dữ liệu bằng SQL (Querying data with SQL):

- **English:** Write SQL queries to extract relevant data from the ChEMBL database. Pay attention to the error you mentioned regarding the `numeric ~ unknown` operator. We need to ensure that we are comparing numeric values correctly.
- **Tiếng Việt:** Viết các truy vấn SQL để trích xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL. Chú ý đến lỗi bạn đã đề cập liên quan đến toán tử `numeric ~ unknown`. Chúng ta cần đảm bảo rằng chúng ta đang so sánh các giá trị số một cách chính xác.

2.3. Xử lý dữ liệu bằng Python và RDKit (Data processing with Python and RDKit):

- **English:** Use RDKit to calculate molecular properties such as molecular weight, logP, and TPSA. Use pandas to store and manipulate the data.
- **Tiếng Việt:** Sử dụng RDKit để tính toán các thuộc tính phân tử như trọng lượng phân tử, logP và TPSA. Sử dụng pandas để lưu trữ và thao tác dữ liệu.

2.4. Xây dựng mô hình (Model building):

- **English:** Use scikit-learn to build predictive models. Address the issue with the `squared=False` parameter in `mean_squared_error` by either updating scikit-learn or using `squared=True` and taking the square root of the result.
- **Tiếng Việt:** Sử dụng scikit-learn để xây dựng các mô hình dự đoán. Giải quyết vấn đề với tham số `squared=False` trong `mean_squared_error` bằng cách cập nhật scikit-learn hoặc sử dụng `squared=True` và lấy căn bậc hai của kết quả.

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

3.1. SQL (Trích xuất dữ liệu từ ChEMBL):

```sql
-- English: Extract 100 rows of activity data for a specific target.
-- Tiếng Việt: Trích xuất 100 dòng dữ liệu hoạt tính cho một mục tiêu cụ thể.
SELECT act.molregno, act.standard_value, act.standard_units, act.standard_type,
md.molsmiles
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.target_chembl_id = 'CHEMBL204' -- Replace with your target of interest
  AND act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0 -- Avoid zero values
  AND act.standard_value::text ~ '^[0-9\.]+$' -- Corrected line: Ensure standard_value
is numeric
LIMIT 100;
```

Giải thích lỗi và cách sửa (Error explanation and fix):

Lỗi `ERROR: operator does not exist: numeric ~ unknown` xảy ra vì PostgreSQL không thể so sánh trực tiếp kiểu dữ liệu `numeric` với một chuỗi (`unknown`). Để khắc phục, chúng ta cần ép kiểu `standard_value` về kiểu `text` và sử dụng toán tử `~` để so sánh với biểu thức chính quy, đảm bảo rằng giá trị chỉ chứa số và dấu chấm.

3.2. Python (Kết nối, xử lý dữ liệu và tính toán thuộc tính RDKit):

```python
# English: Connect to the database, fetch data, and calculate RDKit properties.
# Tiếng Việt: Kết nối đến cơ sở dữ liệu, lấy dữ liệu và tính toán các thuộc tính
RDKit.

import psycopg2
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import os

# Database credentials
db_ip = '192.168.206.136'
db_user = 'rd'
db_pass = 'rd'
db_name = 'chembl_35'

# Base path for data
base_path = '../data'

# SQL query (replace with your query)
sql_query = """
SELECT act.molregno, act.standard_value, act.standard_units, act.standard_type,
md.molsmiles
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
```

```
WHERE act.target_chembl_id = 'CHEMBL204'
  AND act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0
  AND act.standard_value::text ~ '^[0-9\.]+$'
LIMIT 100;
"""

try:
    # Establish connection
    conn = psycopg2.connect(host=db_ip, user=db_user, password=db_pass,
database=db_name)

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

    # Close connection
    conn.close()

    # RDKit calculations
    def calculate_properties(smiles):
        try:
            mol = Chem.MolFromSmiles(smiles)
            if mol is not None:
                mw = Descriptors.MolWt(mol)
                logp = Descriptors.MolLogP(mol)
                tpsa = Descriptors.TPSA(mol)
                return mw, logp, tpsa
            else:
                return None, None, None
        except:
            return None, None, None

    df[['MW', 'LogP', 'TPSA']] = df['molsmiles'].apply(lambda x:
pd.Series(calculate_properties(x)))

    # Save the DataFrame to a CSV file
    csv_file_path = os.path.join(base_path, 'chembl_data.csv')
    df.to_csv(csv_file_path, index=False)
    print(f"Data saved to: {csv_file_path}")

except psycopg2.Error as e:
    print(f"Error connecting to the database: {e}")
except Exception as e:
    print(f"An error occurred: {e}")
```

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

Ví dụ 1: Lọc các hợp chất có trọng lượng phân tử trong một khoảng nhất định (Filtering compounds by molecular weight range):

- **SQL:**

```
-- English: Extract compounds with molecular weight between 200 and 500.
-- Tiếng Việt: Trích xuất các hợp chất có trọng lượng phân tử từ 200 đến 500.
SELECT md.molregno, md.molsmiles
FROM molecule_dictionary md
WHERE md.molregno IN (SELECT act.molregno FROM activities act WHERE act.standard_type
= 'IC50' AND act.standard_units = 'nM')
```

```sql
AND md.mw_freebase BETWEEN 200 AND 500
LIMIT 100;
```

- **Python:**

```python
# English: Filter compounds by molecular weight range using pandas.
# Tiếng Việt: Lọc các hợp chất theo khoảng trọng lượng phân tử sử dụng pandas.
import pandas as pd
# Assuming df is already loaded with data
df_filtered = df[(df['MW'] >= 200) & (df['MW'] <= 500)]
print(df_filtered.head())
```

Ví dụ 2: Tính toán số lượng hợp chất cho mỗi mục tiêu (Calculating the number of compounds per target):

- **SQL:**

```sql
-- English: Count the number of compounds for each target.
-- Tiếng Việt: Đếm số lượng hợp chất cho mỗi mục tiêu.
SELECT target_chembl_id, COUNT(DISTINCT molregno) AS num_compounds
FROM activities
WHERE standard_type = 'IC50' AND standard_units = 'nM'
GROUP BY target_chembl_id
ORDER BY num_compounds DESC
LIMIT 10;
```

- **Python:**

```python
# English: Count the number of compounds for each target using pandas.
# Tiếng Việt: Đếm số lượng hợp chất cho mỗi mục tiêu sử dụng pandas.
import pandas as pd
# Assuming df is already loaded with data containing 'target_chembl_id'
compound_counts = df.groupby('target_chembl_id')['molregno'].nunique()
print(compound_counts.sort_values(ascending=False).head(10))
```

Ví dụ 3: Trích xuất các hợp chất có hoạt tính tốt hơn một ngưỡng nhất định (Extracting compounds with activity better than a certain threshold):

- **SQL:**

```sql
-- English: Extract compounds with IC50 less than 100 nM.
-- Tiếng Việt: Trích xuất các hợp chất có IC50 nhỏ hơn 100 nM.
SELECT act.molregno, act.standard_value, md.molsmiles
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value < 100
LIMIT 100;
```

- **Python:**

```python
# English: Filter compounds with IC50 less than 100 nM using pandas.
# Tiếng Việt: Lọc các hợp chất có IC50 nhỏ hơn 100 nM sử dụng pandas.
import pandas as pd
# Assuming df is already loaded with data containing 'standard_value'
df_active = df[df['standard_value'] < 100]
print(df_active.head())
```

Ví dụ 4: Tính toán thống kê mô tả cho các giá trị hoạt tính (Calculating descriptive statistics for activity values):

- **SQL:**

```sql
-- English: Calculate the average and standard deviation of IC50 values.
-- Tiếng Việt: Tính trung bình và độ lệch chuẩn của các giá trị IC50.
```

```sql
SELECT AVG(standard_value), STDDEV(standard_value)
FROM activities
WHERE standard_type = 'IC50' AND standard_units = 'nM';
```

- **Python:**

```python
# English: Calculate descriptive statistics for IC50 values using pandas.
# Tiếng Việt: Tính thống kê mô tả cho các giá trị IC50 sử dụng pandas.
import pandas as pd
# Assuming df is already loaded with data containing 'standard_value'
print(df['standard_value'].describe())
```

Ví dụ 5: Kết hợp dữ liệu hoạt tính với thông tin mục tiêu (Combining activity data with target information):

- **SQL:**

```sql
-- English: Join activity data with target information to get target names.
-- Tiếng Việt: Kết hợp dữ liệu hoạt tính với thông tin mục tiêu để lấy tên mục tiêu.
SELECT act.molregno, act.standard_value, td.pref_name
FROM activities act
JOIN target_dictionary td ON act.target_chembl_id = td.chembl_id
WHERE act.standard_type = 'IC50' AND act.standard_units = 'nM'
LIMIT 100;
```

- **Python:** (Ví dụ này giả định bạn đã có dữ liệu từ SQL và muốn kết hợp thêm thông tin khác từ một nguồn khác)

```python
# Giả sử bạn có một DataFrame df_targets chứa thông tin mục tiêu
# df_targets có cột 'chembl_id' và 'target_name'
# Kết hợp df (chứa dữ liệu hoạt tính) với df_targets dựa trên 'target_chembl_id'

# Đổi tên cột 'chembl_id' trong df_targets thành 'target_chembl_id' để phù hợp
# df_targets = df_targets.rename(columns={'chembl_id': 'target_chembl_id'})

# Thực hiện merge
# df_merged = pd.merge(df, df_targets, on='target_chembl_id', how='left')

# print(df_merged.head())
```

Lưu ý quan trọng:

- **Phiên bản Scikit-learn:** Nếu bạn không thể cập nhật scikit-learn, hãy sử dụng `mean_squared_error(y_true, y_pred, squared=True)` và lấy căn bậc hai của kết quả để có RMSE (Root Mean Squared Error).
- **Đường dẫn file:** Luôn sử dụng `os.path.join(base_path, 'filename.csv')` để đảm bảo tính tương thích trên các hệ điều hành khác nhau.
- **Giới hạn dữ liệu:** Bắt đầu với 100 dòng dữ liệu như bạn đã đề xuất để giảm tải cho máy. Sau khi mọi thứ hoạt động tốt, bạn có thể tăng dần số lượng dữ liệu.
- **Xử lý lỗi:** Luôn thêm các khối `try...except` để xử lý các lỗi có thể xảy ra trong quá trình kết nối cơ sở dữ liệu, tính toán RDKit hoặc xây dựng mô hình.

Hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án của mình một cách suôn sẻ. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi!