PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Chọn đặc trưng trong mô hình từ ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 này. 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 bắt đầu.

1. Phân tích Mô hình (Analysis Model)

Dự án của bạn tập trung vào việc tận dụng dữ liệu ChEMBL 35 (cơ sở dữ liệu lớn về các phân tử hoạt tính sinh học) kết hợp với RDKit (toolkit mã nguồn mở cho tin học hóa học) để hỗ trợ nghiên cứu và phát triển thuốc. Mô hình phân tích tổng quan sẽ bao gồm các bước sau:

- **Trích xuất dữ liệu từ ChEMBL 35:** Sử dụng SQL để truy vấn dữ liệu cần thiết từ cơ sở dữ liệu PostgreSQL (psql). Dữ liệu này có thể bao gồm thông tin về các hợp chất, hoạt tính sinh học của chúng, các mục tiêu (targets) mà chúng tác động lên, v.v.
- **Tiền xử lý dữ liệu:** Làm sạch và chuẩn hóa dữ liệu. Xử lý các giá trị bị thiếu, loại bỏ các bản ghi trùng lặp, và chuyển đổi dữ liệu thành các định dạng phù hợp cho phân tích.
- **Tính toán đặc trưng phân tử (Molecular Feature Calculation):** Sử dụng RDKit để tính toán các đặc trưng hóa học (descriptors) từ cấu trúc phân tử (SMILES strings). Các đặc trưng này có thể bao gồm trọng lượng phân tử, logP, số lượng liên kết hydro, v.v.
- **Phân tích thống kê và học máy:** Áp dụng các phương pháp thống kê và học máy để khám phá các mối quan hệ giữa cấu trúc phân tử và hoạt tính sinh học. Điều này có thể bao gồm các mô hình hồi quy (regression models) để dự đoán hoạt tính, các mô hình phân loại (classification models) để xác định các chất ức chế tiềm năng, hoặc các phương pháp clustering để nhóm các hợp chất tương tự.
- **Trực quan hóa dữ liệu:** Sử dụng các công cụ trực quan hóa để trình bày kết quả phân tích một cách dễ hiểu. Điều này có thể bao gồm các biểu đồ phân tán (scatter plots), biểu đồ hộp (box plots), hoặc các biểu đồ heatmap để hiển thị mối tương quan giữa các biến.

Song ngữ:

- **English:** Your project focuses on leveraging ChEMBL 35 data (a large database of bioactive molecules) combined with RDKit (an open-source cheminformatics toolkit) to support drug discovery and development. The overall analysis model will include the following steps:
    - **Data Extraction from ChEMBL 35:** Use SQL to query the necessary data from the PostgreSQL (psql) database. This data may include information about compounds, their bioactivities, the targets they act upon, etc.
    - **Data Preprocessing:** Clean and normalize the data. Handle missing values, remove duplicate records, and transform the data into suitable formats for analysis.
    - **Molecular Feature Calculation:** Use RDKit to calculate chemical descriptors from molecular structures (SMILES strings). These features may include molecular weight, logP, the number of hydrogen bonds, etc.
    - **Statistical Analysis and Machine Learning:** Apply statistical and machine learning methods to explore the relationships between molecular structure and bioactivity. This may include regression models to predict activity, classification models to identify potential inhibitors, or clustering methods to group similar compounds.
    - **Data Visualization:** Use visualization tools to present the analysis results in an easily understandable way. This may include scatter plots, box plots, or heatmaps to display correlations between variables.

2. Hướng dẫn song ngữ (Bilingual Instructions)

Bước 1: Kết nối với cơ sở dữ liệu ChEMBL 35 (Connecting to the ChEMBL 35 Database)

- **SQL:** Sử dụng pgAdmin hoặc một công cụ tương tự để kết nối đến cơ sở dữ liệu ChEMBL 35 của bạn. Đảm bảo rằng bạn đã cung cấp đúng thông tin đăng nhập (IP address, username, password, database name).

```
-- Ví dụ kết nối (Example connection - this is not executable SQL)
-- Host: 192.168.206.136
-- User: rd
-- Password: rd
-- Database: chembl_35
```

- **Python:** Sử dụng thư viện `psycopg2` để kết nối đến cơ sở dữ liệu từ Python.

```
import psycopg2
import pandas as pd
import os

# Cấu hình kết nối
host = "192.168.206.136"
database = "chembl_35"
user = "rd"
password = "rd"
port = "5432"

# Hàm kết nối đến cơ sở dữ liệu
def connect_to_db(host, database, user, password, port):
    try:
        conn = psycopg2.connect(host=host, database=database, user=user,
password=password, port=port)
        print("Kết nối thành công!")
        return conn
    except psycopg2.Error as e:
        print(f"Lỗi kết nối: {e}")
        return None

# Kết nối
conn = connect_to_db(host, database, user, password, port)

#Kiểm tra kết nối
if conn:
    conn.close()
```

- **English:**
    - **SQL:** Use pgAdmin or a similar tool to connect to your ChEMBL 35 database. Make sure you have provided the correct credentials (IP address, username, password, database name).
    - **Python:** Use the `psycopg2` library to connect to the database from Python.

Bước 2: Trích xuất dữ liệu (Data Extraction)

- **SQL:** Viết các truy vấn SQL để trích xuất dữ liệu cần thiết. Ví dụ, để lấy thông tin về các hợp chất và hoạt tính của chúng, bạn có thể sử dụng truy vấn sau:

```
SELECT
    md.molregno,
    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)
  AND act.standard_relation = '='  -- Lọc theo quan hệ (ví dụ: =)
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0
LIMIT 100; -- Giới hạn 100 dòng
```

Lưu ý về lỗi `numeric ~ unknown`: Lỗi này thường xảy ra khi bạn cố gắng sử dụng toán tử `~` (regular expression matching) trên một cột kiểu số. Để khắc phục, bạn có thể ép kiểu cột `standard_value` về kiểu text trước khi so sánh. Tuy nhiên, trong trường hợp này, bạn đã lọc `IS NOT NULL` và `> 0` rồi, nên có thể bỏ qua dòng `AND act.standard_value ~ '^[0-9\.]+$'` để tránh lỗi.

- **Python:** Sử dụng thư viện `pandas` để đọc dữ liệu từ kết quả truy vấn SQL.

```
import pandas as pd

# Thực thi truy vấn SQL và đọc vào DataFrame
def execute_query(conn, query):
    try:
        df = pd.read_sql_query(query, conn)
        return df
    except Exception as e:
        print(f"Lỗi truy vấn: {e}")
        return None

# Truy vấn SQL (ví dụ)
query = """
SELECT
    md.molregno,
    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_relation = '='
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0
LIMIT 100;
"""

# Thực thi truy vấn
df = execute_query(conn, query)

# In ra DataFrame
```

```
if df is not None:
    print(df.head())
```

- **English:**
    - **SQL:** Write SQL queries to extract the necessary data. For example, to retrieve information about compounds and their activities, you can use the following query:
    - **Python:** Use the pandas library to read data from the SQL query results.

Bước 3: Tiền xử lý dữ liệu (Data Preprocessing)

- **Python:** Sử dụng pandas để làm sạch và chuẩn hóa dữ liệu.

```
# Xử lý giá trị bị thiếu
df = df.dropna()

# Loại bỏ các bản ghi trùng lặp
df = df.drop_duplicates()

# Chuyển đổi đơn vị (nếu cần)
# Ví dụ: chuyển đổi tất cả các giá trị IC50 về nM
def convert_to_nM(value, unit):
    if unit == 'nM':
        return value
    elif unit == 'uM':
        return value * 1000
    else:
        return None

df['standard_value_nM'] = df.apply(lambda row:
convert_to_nM(row['standard_value'], row['standard_units']), axis=1)
df = df.dropna(subset=['standard_value_nM'])

print(df.head())
```

- **English:**
    - **Python:** Use pandas to clean and normalize the data.

Bước 4: Tính toán đặc trưng phân tử (Molecular Feature Calculation)

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

```
from rdkit import Chem
from rdkit.Chem import Descriptors

# Hàm tính toán các đặc trưng phân tử
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    descriptors['NumHDonors'] = Descriptors.NumHDonors(mol)
    descriptors['NumHAcceptors'] = Descriptors.NumHAcceptors(mol)
    return descriptors

# Áp dụng hàm tính toán đặc trưng cho mỗi SMILES
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)
```

```
# Tạo các cột riêng biệt cho từng đặc trưng
df = pd.concat([df.drop(['descriptors'], axis=1),
df['descriptors'].apply(pd.Series)], axis=1)

print(df.head())
```

- **English:**
    - **Python:** Use RDKit to calculate molecular features.

Bước 5: Phân tích và mô hình hóa (Analysis and Modeling)

- **Python:** Sử dụng các thư viện như `scikit-learn` để xây dựng các mô hình học máy.

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

# Chuẩn bị dữ liệu
X = df[['MolWt', 'LogP', 'NumHDonors', 'NumHAcceptors']].dropna()
y = df['standard_value_nM'].dropna()

# 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)
print(f'Mean Squared Error: {mse}')
```

Lưu ý về lỗi `squared=False`: Nếu bạn đang sử dụng phiên bản `scikit-learn` cũ, hãy bỏ tham số `squared=False` khỏi hàm `mean_squared_error`.

- **English:**
    - **Python:** Use libraries like `scikit-learn` to build machine learning models.

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

Xem các đoạn code ở trên trong phần "Hướng dẫn song ngữ".

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

Dưới đây là 5 ví dụ code SQL và Python khác nhau để minh họa các thao tác khác nhau trên dữ liệu ChEMBL 35.

Ví dụ 1: Lọc các hợp chất có hoạt tính IC50 nhỏ hơn 100 nM

- **SQL:**

```
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value
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_relation = '='
  AND act.standard_units = 'nM'
  AND act.standard_value < 100
LIMIT 10;
```

- **Python:**

```
query = """
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value
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_relation = '='
  AND act.standard_units = 'nM'
  AND act.standard_value < 100
LIMIT 10;
"""

df = execute_query(conn, query)
if df is not None:
    print(df)
```

Ví dụ 2: Tính số lượng hợp chất cho mỗi mục tiêu (target)

- **SQL:**

```
SELECT
    td.chembl_id,
    td.pref_name,
    COUNT(DISTINCT act.molregno) AS num_compounds
FROM
    target_dictionary td
JOIN
    activities act ON td.tid = act.tid
GROUP BY
    td.chembl_id, td.pref_name
ORDER BY
    num_compounds DESC
LIMIT 10;
```

- **Python:**

```
query = """
SELECT
    td.chembl_id,
    td.pref_name,
    COUNT(DISTINCT act.molregno) AS num_compounds
FROM
```

```
    target_dictionary td
JOIN
    activities act ON td.tid = act.tid
GROUP BY
    td.chembl_id, td.pref_name
ORDER BY
    num_compounds DESC
LIMIT 10;
"""

df = execute_query(conn, query)
if df is not None:
    print(df)
```

Ví dụ 3: Lấy thông tin về các protein mục tiêu và trình tự của chúng

- **SQL:**

```
SELECT
    td.chembl_id,
    td.pref_name,
    sp.sequence
FROM
    target_dictionary td
JOIN
    target_components tc ON td.tid = tc.tid
JOIN
    component_sequences sp ON tc.component_id = sp.component_id
WHERE
    td.target_type = 'SINGLE PROTEIN'
LIMIT 10;
```

- **Python:**

```
query = """
SELECT
    td.chembl_id,
    td.pref_name,
    sp.sequence
FROM
    target_dictionary td
JOIN
    target_components tc ON td.tid = tc.tid
JOIN
    component_sequences sp ON tc.component_id = sp.component_id
WHERE
    td.target_type = 'SINGLE PROTEIN'
LIMIT 10;
"""

df = execute_query(conn, query)
if df is not None:
    print(df)
```

Ví dụ 4: Tính toán các đặc trưng phân tử và lưu vào DataFrame

- **Python:** (Đã được trình bày ở trên trong phần "Tính toán đặc trưng phân tử")

Ví dụ 5: Xây dựng mô hình hồi quy tuyến tính (Linear Regression Model)

- **Python:** (Đã được trình bày ở trên trong phần "Phân tích và mô hình hóa")

Hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án của mình một cách hiệu quả. Chúc bạn thành công!