PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Ước lượng thời gian ra thị trường từ ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 này. Dựa trên thông tin bạn cung cấp, chúng ta sẽ cùng nhau xây dựng quy trình làm việc hiệu quả, giải quyết các vấn đề phát sinh và tạo ra những kết quả có giá trị cho nghiên cứu phát triển thuốc.

Tổng quan dự án:

Dự án của bạn tập trung vào việc phân tích dữ liệu ChEMBL 35, một cơ sở dữ liệu lớn chứa thông tin về các phân tử sinh học hoạt tính. Bạn sẽ sử dụng PostgreSQL để truy vấn dữ liệu, RDKit để xử lý cấu trúc hóa học và Jupyter Notebook để phân tích và trực quan hóa dữ liệu. Quy trình làm việc tuân theo khuôn khổ MLOps, đảm bảo tính nhất quán và khả năng tái sản xuất.

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

Mục tiêu chính của bạn là gì? Dựa trên mã chủ đề "Topic_CheMBL_35_44", tôi đoán rằng bạn đang muốn tập trung vào một mục tiêu cụ thể, ví dụ:

- **Dự đoán hoạt tính của các hợp chất:** Xây dựng mô hình dự đoán hoạt tính của các hợp chất mới dựa trên cấu trúc hóa học và các đặc tính khác.
- **Tìm kiếm các hợp chất tiềm năng:** Xác định các hợp chất có hoạt tính mong muốn dựa trên truy vấn cấu trúc hoặc các tiêu chí khác.
- **Phân tích mối quan hệ cấu trúc-hoạt tính (SAR):** Tìm hiểu mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học để tối ưu hóa các hợp chất.
- **Xây dựng mô hình QSAR/QSPR:** Phát triển các mô hình định lượng mối quan hệ cấu trúc-hoạt tính (QSAR) hoặc cấu trúc-tính chất (QSPR) để dự đoán hoạt tính hoặc tính chất của các hợp chất.

Để đạt được mục tiêu này, bạn có thể sử dụng các kỹ thuật sau:

- **Truy vấn SQL:** Sử dụng SQL để trích xuất dữ liệu cần thiết từ cơ sở dữ liệu ChEMBL, bao gồm thông tin về hợp chất, hoạt tính, mục tiêu, v.v.
- **Xử lý cấu trúc hóa học với RDKit:** Sử dụng RDKit để tính toán các descriptor hóa học (ví dụ: trọng lượng phân tử, logP, số lượng liên kết) từ cấu trúc SMILES của các hợp chất.
- **Phân tích dữ liệu và trực quan hóa:** Sử dụng Python, Pandas, NumPy và Matplotlib/Seaborn để khám phá dữ liệu, làm sạch dữ liệu và tạo ra các biểu đồ và đồ thị trực quan.
- **Xây dựng mô hình học máy:** Sử dụng Scikit-learn hoặc các thư viện khác để xây dựng các mô hình học máy (ví dụ: hồi quy tuyến tính, random forest, mạng nơ-ron) để dự đoán hoạt tính hoặc tính chất của các hợp chất.

2. Hướng dẫn song ngữ (Bilingual Guidance):

Dưới đây là một số hướng dẫn song ngữ để giúp bạn trong quá trình thực hiện dự án:

- **SQL:**
    - "SELECT * FROM table_name WHERE condition;" - "Chọn tất cả các cột từ bảng 'table_name' nơi điều kiện là đúng."
    - "CREATE TABLE table_name (column_name data_type, ...);" - "Tạo một bảng mới có tên 'table_name' với các cột và kiểu dữ liệu được chỉ định."
    - "INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);" - "Chèn một hàng mới vào bảng 'table_name' với các giá trị được chỉ định cho các cột."

- **Python:**
  - "import library_name" - "Nhập thư viện 'library_name' để sử dụng các hàm và lớp của nó."
  - "variable_name = value" - "Gán giá trị cho biến 'variable_name'."
  - "def function_name(parameters):" - "Định nghĩa một hàm mới có tên 'function_name' với các tham số được chỉ định."
  - "for item in list_name:" - "Lặp qua từng phần tử trong danh sách 'list_name'."

3. Code SQL, Python (English and Vietnamese):

SQL (English):

```sql
-- Get 100 compounds with their SMILES and standard value
SELECT md.molregno,
       cs.canonical_smiles,
       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 cs.canonical_smiles IS NOT NULL
LIMIT 100;
```

SQL (Tiếng Việt):

```sql
-- Lấy 100 hợp chất cùng với SMILES và giá trị chuẩn của chúng
SELECT md.molregno,
       cs.canonical_smiles,
       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 cs.canonical_smiles IS NOT NULL
LIMIT 100;
```

Python (English):

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

# Database credentials
db_host = "192.168.206.136"
db_user = "rd"
db_pass = "rd"
db_name = "chembl_35"

# Establish connection
conn = psycopg2.connect(host=db_host, user=db_user, password=db_pass,
database=db_name)
```

```
# SQL query to fetch data
sql_query = """
SELECT md.molregno,
       cs.canonical_smiles,
       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 cs.canonical_smiles IS NOT NULL
LIMIT 100;
"""

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

# Close connection
conn.close()

# Function to calculate molecular weight using RDKit
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Apply the function to the DataFrame
df['molecular_weight'] = df['canonical_smiles'].apply(calculate_mw)

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

Python (Tiếng Việt):

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

# Thông tin đăng nhập cơ sở dữ liệu
db_host = "192.168.206.136"
db_user = "rd"
db_pass = "rd"
db_name = "chembl_35"

# Thiết lập kết nối
conn = psycopg2.connect(host=db_host, user=db_user, password=db_pass,
database=db_name)

# Truy vấn SQL để lấy dữ liệu
sql_query = """
SELECT md.molregno,
       cs.canonical_smiles,
       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 cs.canonical_smiles IS NOT NULL
LIMIT 100;
"""

# Đọc dữ liệu vào một DataFrame Pandas
df = pd.read_sql_query(sql_query, conn)

# Đóng kết nối
conn.close()

# Hàm tính toán trọng lượng phân tử sử dụng RDKit
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Áp dụng hàm cho DataFrame
df['molecular_weight'] = df['canonical_smiles'].apply(calculate_mw)

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

Giải quyết lỗi:

- **Lỗi SQL: `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`**

  Lỗi này xảy ra vì bạn đang cố gắng sử dụng toán tử ~ (regular expression match) trên một cột kiểu số (numeric). Để khắc phục, bạn có thể chuyển đổi cột `standard_value` sang kiểu text trước khi so sánh:

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

  Tuy nhiên, việc lọc các giá trị không phải số có thể không cần thiết nếu bạn đã lọc `act.standard_value IS NOT NULL`. Hãy đảm bảo kiểu dữ liệu của `act.standard_value` là phù hợp.

- **Lỗi Python: `phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean_squared_error`**

  Nâng cấp phiên bản scikit-learn của bạn lên phiên bản mới hơn bằng lệnh:

  ```
  pip install scikit-learn --upgrade
  ```

  Hoặc, nếu bạn không thể nâng cấp, hãy bỏ tham số `squared=False` và tính căn bậc hai của kết quả sau:

  ```
  from sklearn.metrics import mean_squared_error
  mse = mean_squared_error(y_true, y_predicted)
  rmse = mse**0.5  # Calculate RMSE
  ```

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

Dưới đây là 5 ví dụ về các tác vụ bạn có thể thực hiện với dữ liệu ChEMBL 35:

Ví dụ 1: Lọc các hợp chất có IC50 dưới 100nM (SQL & Python)

SQL:

```sql
SELECT md.molregno, 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_units = 'nM'
  AND act.standard_value <= 100
LIMIT 100;
```

Python:

```python
import pandas as pd
import psycopg2

# Database credentials (replace with your actual credentials)
db_host = "192.168.206.136"
db_user = "rd"
db_pass = "rd"
db_name = "chembl_35"

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

# SQL query to fetch data
sql_query = """
SELECT md.molregno, 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_units = 'nM'
  AND act.standard_value <= 100
LIMIT 100;
"""

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

# Close connection
conn.close()

print(df.head())
```

Ví dụ 2: Tính toán LogP cho các hợp chất (SQL & Python)

(Ví dụ này tập trung vào phần Python, vì LogP được tính bằng RDKit)

```python
import pandas as pd
import psycopg2
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Descriptors

# Database credentials (replace with your actual credentials)
```

```
db_host = "192.168.206.136"
db_user = "rd"
db_pass = "rd"
db_name = "chembl_35"

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

# SQL query to fetch data
sql_query = """
SELECT md.molregno, cs.canonical_smiles
FROM molecule_dictionary md
JOIN compound_structures cs ON md.molregno = cs.molregno
LIMIT 100;
"""

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

# Close connection
conn.close()

# Function to calculate LogP using RDKit
def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolLogP(mol)
    else:
        return None

# Apply the function to the DataFrame
df['logp'] = df['canonical_smiles'].apply(calculate_logp)

print(df.head())
```

Ví dụ 3: Tìm kiếm các hợp chất tương tự (SQL & Python)

(Ví dụ này tập trung vào phần Python, vì tìm kiếm tương tự dựa trên fingerprint của cấu trúc)

```
import pandas as pd
import psycopg2
from rdkit import Chem
from rdkit.Chem import AllChem
import numpy as np

# Database credentials (replace with your actual credentials)
db_host = "192.168.206.136"
db_user = "rd"
db_pass = "rd"
db_name = "chembl_35"

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

# SQL query to fetch data
sql_query = """
SELECT md.molregno, cs.canonical_smiles
FROM molecule_dictionary md
```

```
JOIN compound_structures cs ON md.molregno = cs.molregno
LIMIT 100;
"""

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

# Close connection
conn.close()

# Define a reference compound (replace with a ChEMBL ID or SMILES)
reference_smiles = 'Cc1ccccc1C(=O)O'  # Example: Aspirin
reference_mol = Chem.MolFromSmiles(reference_smiles)
reference_fp = AllChem.GetMorganFingerprintAsBitVect(reference_mol, 2, nBits=2048)

# Function to calculate Tanimoto similarity
def calculate_tanimoto(smiles, reference_fp):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
        return DataStructs.TanimotoSimilarity(reference_fp, fp)
    else:
        return 0.0

from rdkit import DataStructs
# Calculate Tanimoto similarity for all compounds in the DataFrame
df['tanimoto_similarity'] = df['canonical_smiles'].apply(lambda x:
calculate_tanimoto(x, reference_fp))

# Filter compounds with a similarity score above a threshold (e.g., 0.7)
similar_compounds = df[df['tanimoto_similarity'] > 0.7]

print(similar_compounds)
```

Ví dụ 4: Thống kê số lượng hợp chất cho mỗi mục tiêu (SQL)

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

(Ví dụ này chỉ có SQL vì nó là một truy vấn thống kê đơn giản)

Ví dụ 5: Tạo mô hình dự đoán IC50 đơn giản (Python)

```
import pandas as pd
import psycopg2
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 numpy as np

# Database credentials (replace with your actual credentials)
db_host = "192.168.206.136"
db_user = "rd"
db_pass = "rd"
db_name = "chembl_35"
```

```
# Establish connection
conn = psycopg2.connect(host=db_host, user=db_user, password=db_pass,
database=db_name)

# SQL query to fetch data
sql_query = """
SELECT md.molregno, 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_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND cs.canonical_smiles IS NOT NULL
LIMIT 100;
"""

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

# Close connection
conn.close()

# Function to calculate molecular weight using RDKit
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Apply the function to the DataFrame
df['molecular_weight'] = df['canonical_smiles'].apply(calculate_mw)

# Drop rows with missing molecular weight
df = df.dropna(subset=['molecular_weight', 'standard_value'])

# Prepare data for the model
X = df[['molecular_weight']]
y = np.log10(df['standard_value']) # Log transform IC50 values

# 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)

# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print(f"Root Mean Squared Error: {rmse}")
```

Lưu ý quan trọng:

- **Thay thế thông tin đăng nhập cơ sở dữ liệu:** Đảm bảo thay thế các giá trị `db_host`, `db_user`, `db_pass` và `db_name` bằng thông tin đăng nhập thực tế của bạn.
- **Xử lý lỗi:** Code trên chỉ là ví dụ và có thể cần thêm xử lý lỗi để đảm bảo tính ổn định.
- **Làm sạch dữ liệu:** Dữ liệu ChEMBL có thể chứa các giá trị bị thiếu hoặc không hợp lệ. Hãy đảm bảo làm sạch dữ liệu trước khi sử dụng nó cho phân tích hoặc mô hình hóa.
- **Lựa chọn descriptor:** Trọng lượng phân tử chỉ là một descriptor đơn giản. Để xây dựng các mô hình dự đoán tốt hơn, hãy sử dụng nhiều descriptor hơn và xem xét các phương pháp lựa chọn descriptor.
- **Đánh giá mô hình:** Đánh giá mô hình của bạn bằng các bộ dữ liệu kiểm tra độc lập để đảm bảo khả năng khái quát hóa.

Tôi hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án phân tích dữ liệu ChEMBL 35 của mình. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi!