PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Dự đoán khả năng thành công lâm sàng

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit này. Với vai trò là một chuyên gia nghiên cứu và phát triển dược học, tôi hiểu rõ tầm quan trọng của việc khai thác dữ liệu từ các nguồn như ChEMBL để tăng tốc quá trình phát triển thuốc.

Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu SQL và Python, cùng các ví dụ bạn có thể tham khảo:

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

Dự án của bạn tập trung vào việc sử dụng dữ liệu ChEMBL 35, một cơ sở dữ liệu hóa học sinh học lớn chứa thông tin về các phân tử hoạt tính sinh học. Kết hợp với RDKit, một bộ công cụ hóa tin học mạnh mẽ, bạn có thể thực hiện nhiều phân tích khác nhau để khám phá các mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học.

Các bước phân tích chính:

- **Thu thập và tiền xử lý dữ liệu (Data Acquisition and Preprocessing):**
    - Sử dụng SQL để truy vấn và trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL 35.
    - Làm sạch và chuẩn hóa dữ liệu, xử lý các giá trị thiếu và loại bỏ các bản ghi không hợp lệ.
    - Chuyển đổi dữ liệu sang định dạng phù hợp để phân tích bằng RDKit và các công cụ khác.
- **Tính toán các đặc trưng hóa học (Chemical Feature Calculation):**
    - Sử dụng RDKit để tính toán các đặc trưng hóa học của các phân tử, chẳng hạn như trọng lượng phân tử, logP, số lượng vòng, số lượng nguyên tử, v.v.
    - Tính toán các fingerprint (ECFP, MACCS) để biểu diễn cấu trúc phân tử một cách số hóa.
- **Phân tích tương quan (Correlation Analysis):**
    - Tìm kiếm mối tương quan giữa các đặc trưng hóa học và hoạt tính sinh học.
    - Sử dụng các phương pháp thống kê như hệ số tương quan Pearson hoặc Spearman để đánh giá mức độ tương quan.
- **Xây dựng mô hình dự đoán (Predictive Model Building):**
    - Sử dụng các thuật toán học máy như hồi quy tuyến tính, Support Vector Machines (SVM), Random Forest, hoặc mạng nơ-ron để xây dựng mô hình dự đoán hoạt tính sinh học dựa trên các đặc trưng hóa học.
    - Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số như RMSE, R-squared, AUC, v.v.
- **Phân tích SAR (Structure-Activity Relationship Analysis):**
    - Xác định các nhóm thế hoặc cấu trúc con quan trọng ảnh hưởng đến hoạt tính sinh học.
    - Sử dụng các kỹ thuật trực quan hóa dữ liệu để khám phá các mối quan hệ SAR.

2. Hướng dẫn song ngữ (Bilingual Instructions)

a. Kết nối đến cơ sở dữ liệu ChEMBL 35 (Connecting to the ChEMBL 35 Database):

- **Tiếng Anh:** Use the provided credentials (IP address, username, password, database name) to establish a connection to the ChEMBL 35 database using a SQL client like pgAdmin or a Python library like `psycopg2`.

- **Tiếng Việt:** Sử dụng thông tin đăng nhập đã cung cấp (địa chỉ IP, tên người dùng, mật khẩu, tên cơ sở dữ liệu) để thiết lập kết nối đến cơ sở dữ liệu ChEMBL 35 bằng một trình khách SQL như pgAdmin hoặc một thư viện Python như psycopg2.

b. Truy vấn dữ liệu bằng SQL (Querying Data with SQL):

- **Tiếng Anh:** Write SQL queries to extract the desired data from the ChEMBL 35 database. Focus on selecting relevant columns such as molecule IDs, SMILES strings, and activity values.
- **Tiếng Việt:** Viết các truy vấn SQL để trích xuất dữ liệu mong muốn từ cơ sở dữ liệu ChEMBL 35. Tập trung vào việc chọn các cột liên quan như ID phân tử, chuỗi SMILES và giá trị hoạt tính.

c. Tiền xử lý dữ liệu bằng Python và RDKit (Data Preprocessing with Python and RDKit):

- **Tiếng Anh:** Use Python libraries like Pandas and RDKit to read the data, clean it, and calculate molecular descriptors.
- **Tiếng Việt:** Sử dụng các thư viện Python như Pandas và RDKit để đọc dữ liệu, làm sạch dữ liệu và tính toán các descriptor phân tử.

d. Xây dựng mô hình học máy (Building Machine Learning Models):

- **Tiếng Anh:** Use scikit-learn or other machine learning libraries to build models that predict activity based on molecular descriptors.
- **Tiếng Việt:** Sử dụng scikit-learn hoặc các thư viện học máy khác để xây dựng các mô hình dự đoán hoạt tính dựa trên các descriptor phân tử.

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

a. SQL (Ví dụ lấy 100 dòng dữ liệu):

```sql
-- Lấy thông tin cơ bản về hoạt tính sinh học và cấu trúc phân tử
SELECT
    act.molregno,  -- ID của phân tử
    cmp.chembl_id, -- ID ChEMBL của phân tử
    cmp.canonical_smiles,  -- Chuỗi SMILES biểu diễn cấu trúc phân tử
    act.standard_type,  -- Loại hoạt tính (ví dụ: IC50, Ki)
    act.standard_value, -- Giá trị hoạt tính
    act.standard_units  -- Đơn vị của giá trị hoạt tính
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
WHERE
    act.standard_type = 'IC50'  -- Lọc theo loại hoạt tính IC50
    AND act.standard_units = 'nM'  -- Lọc theo đơn vị nM
    AND act.standard_value IS NOT NULL  -- Loại bỏ các giá trị NULL
    --Sửa lỗi:
    AND act.standard_value::text ~ '^[0-9\.]+$' -- Đảm bảo giá trị là số
LIMIT 100;
```

b. Python (Sử dụng RDKit và scikit-learn):

```python
import os
import pandas as pd
from rdkit import Chem
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 (thay đổi nếu cần)
base_path = "."  # hoặc os.getcwd() nếu bạn muốn thư mục hiện tại

# 1. Đọc dữ liệu từ file CSV
data_path = os.path.join(base_path, "data", "chembl_ic50_100.csv") #Sửa lại đường dẫn
cho đúng
df = pd.read_csv(data_path)

# 2. Tiền xử lý dữ liệu và tính toán descriptor
def calculate_descriptors(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None
        descriptors = {}
        descriptors["MolWt"] = Descriptors.MolWt(mol)
        descriptors["LogP"] = Descriptors.MolLogP(mol)
        descriptors["HBA"] = Descriptors.NumHAcceptors(mol)
        descriptors["HBD"] = Descriptors.NumHDonors(mol)
        return descriptors
    except:
        return None

# Áp dụng hàm tính descriptor
df["descriptors"] = df["canonical_smiles"].apply(calculate_descriptors)

# Loại bỏ các hàng có descriptor bị lỗi
df = df.dropna(subset=["descriptors"]).reset_index(drop=True)

# Chuyển đổi descriptor thành DataFrame riêng
descriptors_df = pd.json_normalize(df["descriptors"])

# Gộp DataFrame descriptor vào DataFrame chính
df = pd.concat([df, descriptors_df], axis=1)

# Chuyển đổi giá trị IC50 sang dạng số và logarit
df = df.dropna(subset=["standard_value"]).reset_index(drop=True)
df["standard_value"] = pd.to_numeric(df["standard_value"], errors='coerce')
df = df.dropna(subset=["standard_value"]).reset_index(drop=True) #Loại bỏ Nan sau
convert
df["pIC50"] = -np.log10(df["standard_value"] / 1e9)  # Chuyển đổi nM sang M và tính
-log10

# 3. Chuẩn bị dữ liệu cho mô hình
X = df[["MolWt", "LogP", "HBA", "HBD"]]  # Chọn các descriptor làm features
y = df["pIC50"]  # Chọn pIC50 làm target

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X = scaler.fit_transform(X)

# 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)

# 4. Xây dựng và huấn luyện mô hình
model = LinearRegression()
model.fit(X_train, y_train)
```

```python
# 5. Đánh giá mô hình
y_pred = model.predict(X_test)
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. Ví dụ (Examples)

Dưới đây là 5 ví dụ về các truy vấn SQL và đoạn code Python bạn có thể sử dụng:

a. Ví dụ 1: Lấy các hợp chất có IC50 < 100 nM cho một mục tiêu cụ thể (ví dụ: CHEMBL205)

```sql
-- SQL
SELECT
    cmp.chembl_id,
    cmp.canonical_smiles,
    act.standard_value
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value <= 100
    AND act.target_chembl_id = 'CHEMBL205' -- Thay đổi target_chembl_id nếu cần
    AND act.standard_value::text ~ '^[0-9\.]+$'
LIMIT 100;
```

```python
# Python
import pandas as pd
from rdkit import Chem

def filter_compounds(df, max_ic50=100):
  """Lọc các hợp chất có IC50 <= max_ic50 từ DataFrame."""
  df = df[df['standard_value'] <= max_ic50].reset_index(drop=True)
  return df

# Sử dụng hàm filter_compounds sau khi đã đọc dữ liệu từ SQL
# Ví dụ:
# filtered_df = filter_compounds(df, max_ic50=100)
```

b. Ví dụ 2: Tính toán TPSA (Diện tích bề mặt cực đại) cho một tập hợp các SMILES

```python
# Python
from rdkit import Chem
from rdkit.Chem import rdMolDescriptors

def calculate_tpsa(smiles):
    """Tính toán TPSA cho một phân tử dựa trên chuỗi SMILES."""
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        return rdMolDescriptors.CalcTPSA(mol)
    else:
        return None

# Áp dụng hàm calculate_tpsa cho cột SMILES trong DataFrame
# Ví dụ:
# df['tpsa'] = df['canonical_smiles'].apply(calculate_tpsa)
```

c. Ví dụ 3: Tạo fingerprint MACCS cho các phân tử

```python
# Python
from rdkit import Chem
from rdkit.Chem import AllChem

def generate_maccs_fingerprint(smiles):
    """Tạo fingerprint MACCS cho một phân tử từ chuỗi SMILES."""
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        fp = AllChem.GetMACCSKeysFingerprint(mol)
        return fp.ToBitString()  # Trả về fingerprint dưới dạng chuỗi bit
    else:
        return None

# Ví dụ:
# df['maccs_fp'] = df['canonical_smiles'].apply(generate_maccs_fingerprint)
```

d. Ví dụ 4: Lấy các hợp chất có cùng khung scaffold

```sql
--Lấy các hợp chất có khung scaffold giống với một hợp chất cụ thể (ví dụ: CHEMBL121)
SELECT
    cmp.chembl_id,
    cmp.canonical_smiles
FROM
    molecule_dictionary cmp
WHERE
    cmp.standard_inchi_key LIKE (
        SELECT
            SUBSTR(standard_inchi_key, 1, 14) || '%'
        FROM
            molecule_dictionary
        WHERE
            chembl_id = 'CHEMBL121'
    )
LIMIT 100;
```

e. Ví dụ 5: Trực quan hóa kết quả dự đoán mô hình

```python
import matplotlib.pyplot as plt

# Giả sử bạn đã có y_test và y_pred từ mô hình
plt.scatter(y_test, y_pred)
plt.xlabel("Giá trị thực tế (pIC50)")
plt.ylabel("Giá trị dự đoán (pIC50)")
plt.title("Giá trị thực tế so với giá trị dự đoán")
plt.show()
```

Lưu ý quan trọng:

- **Lỗi SQL:** Lỗi "operator does not exist: numeric ~ unknown" xảy ra do bạn đang cố gắng so sánh một cột kiểu số (numeric) với một chuỗi (unknown). Để khắc phục, hãy đảm bảo rằng cả hai vế của phép so sánh đều cùng kiểu dữ liệu. Bạn có thể sử dụng `act.standard_value::text ~ '^[0-9\.]+$'` để chuyển đổi cột `standard_value` sang kiểu text trước khi so sánh.
- **Phiên bản scikit-learn:** Nếu bạn gặp lỗi liên quan đến tham số `squared=False` trong hàm `mean_squared_error`, hãy cập nhật phiên bản scikit-learn của bạn lên phiên bản mới nhất hoặc loại bỏ tham số `squared=False` nếu không cần thiết (phiên bản cũ mặc định trả về MSE, không phải RMSE).
- **Đường dẫn:** Hãy chắc chắn rằng bạn đã thay đổi đường dẫn trong các đoạn code trên để phù hợp với cấu trúc thư mục dự án của bạn.

- **Giới hạn dữ liệu:** Vì lý do hiệu suất, tôi đã giới hạn số lượng bản ghi được truy vấn và xử lý trong các ví dụ trên. Bạn có thể điều chỉnh các giới hạn này tùy thuộc vào nhu cầu của bạn.

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