PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Dự đoán mục tiêu cho các phân tử không rõ target

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Với vai trò là một chuyên gia R&D dược học, tôi sẽ cung cấp cho bạn hướng dẫn chi tiết, code mẫu và giải thích cặn kẽ để giúp bạn đạt được mục tiêu.

1. Phân Tích Mô Hình (Analysis of the Model)

Dựa trên thông tin bạn cung cấp, chúng ta sẽ xây dựng một quy trình làm việc (workflow) phân tích dữ liệu ChEMBL 35 theo các bước sau:

- **Bước 1: Truy Vấn Dữ Liệu từ PostgreSQL (Data Retrieval from PostgreSQL)**: Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35 trên máy chủ PostgreSQL của bạn. Chúng ta sẽ tập trung vào việc lấy dữ liệu cần thiết cho phân tích cấu trúc-hoạt tính (SAR) hoặc các mô hình hóa khác.
- **Bước 2: Tiền Xử Lý Dữ Liệu (Data Preprocessing)**: Sử dụng RDKit để xử lý các phân tử, tính toán các descriptor phân tử (ví dụ: tính chất vật lý, fingerprint), và làm sạch dữ liệu (ví dụ: loại bỏ các giá trị không hợp lệ).
- **Bước 3: Phân Tích Thống Kê và Mô Hình Hóa (Statistical Analysis and Modeling)**: Sử dụng các thư viện như scikit-learn để xây dựng các mô hình dự đoán hoạt tính (ví dụ: hồi quy tuyến tính, random forest) dựa trên các descriptor phân tử.
- **Bước 4: Đánh Giá và Trực Quan Hóa Kết Quả (Evaluation and Visualization)**: Đánh giá hiệu năng của mô hình bằng các độ đo phù hợp (ví dụ: RMSE, R^2) và trực quan hóa kết quả để hiểu rõ hơn về mối quan hệ giữa cấu trúc và hoạt tính.

2. Hướng Dẫn Song Ngữ (Bilingual Guidance)

Dưới đây là hướng dẫn chi tiết bằng cả tiếng Anh và tiếng Việt cho từng bước:

Step 1: Data Retrieval from PostgreSQL (Bước 1: Truy Vấn Dữ Liệu từ PostgreSQL)

- **English:**
  - Connect to the ChEMBL 35 database using your credentials.
  - Write SQL queries to extract relevant data, such as compound structures (canonical_smiles) and activity values (standard_value, standard_type).
  - Handle potential errors, such as filtering out invalid activity values.
- **Tiếng Việt:**
  - Kết nối đến cơ sở dữ liệu ChEMBL 35 sử dụng thông tin đăng nhập của bạn.
  - Viết các truy vấn SQL để trích xuất dữ liệu liên quan, chẳng hạn như cấu trúc hợp chất (canonical_smiles) và giá trị hoạt tính (standard_value, standard_type).
  - Xử lý các lỗi tiềm ẩn, chẳng hạn như lọc bỏ các giá trị hoạt tính không hợp lệ.

Step 2: Data Preprocessing (Bước 2: Tiền Xử Lý Dữ Liệu)

- **English:**
  - Use RDKit to read SMILES strings and generate molecular objects.
  - Calculate molecular descriptors using RDKit functions (e.g., MolWt, LogP, TPSA).
  - Handle errors such as invalid SMILES strings.
  - Convert activity values to a consistent scale (e.g., pIC50).

- **Tiếng Việt:**
    - Sử dụng RDKit để đọc chuỗi SMILES và tạo các đối tượng phân tử.
    - Tính toán các descriptor phân tử bằng các hàm RDKit (ví dụ: MolWt, LogP, TPSA).
    - Xử lý các lỗi như chuỗi SMILES không hợp lệ.
    - Chuyển đổi giá trị hoạt tính về một thang đo thống nhất (ví dụ: pIC50).

Step 3: Statistical Analysis and Modeling (Bước 3: Phân Tích Thống Kê và Mô Hình Hóa)

- **English:**
    - Use scikit-learn to build machine learning models.
    - Split data into training and test sets.
    - Train models such as linear regression, random forest, or support vector machines.
    - Tune hyperparameters using cross-validation.
- **Tiếng Việt:**
    - Sử dụng scikit-learn để xây dựng các mô hình học máy.
    - Chia dữ liệu thành tập huấn luyện và tập kiểm tra.
    - Huấn luyện các mô hình như hồi quy tuyến tính, random forest, hoặc máy vectơ hỗ trợ.
    - Điều chỉnh các siêu tham số bằng cách sử dụng kiểm định chéo.

Step 4: Evaluation and Visualization (Bước 4: Đánh Giá và Trực Quan Hóa Kết Quả)

- **English:**
    - Evaluate model performance using metrics such as RMSE, R^2, or AUC.
    - Visualize results using Matplotlib or Seaborn to understand the relationship between molecular descriptors and activity.
- **Tiếng Việt:**
    - Đánh giá hiệu năng của mô hình bằng các độ đo như RMSE, R^2, hoặc AUC.
    - Trực quan hóa kết quả bằng Matplotlib hoặc Seaborn để hiểu rõ hơn về mối quan hệ giữa các descriptor phân tử và hoạt tính.

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

SQL (để truy vấn dữ liệu từ ChEMBL 35):

```sql
-- Lấy 100 dòng dữ liệu từ bảng 'activities' và 'molecule_dictionary'
SELECT md.chembl_id,
       md.canonical_smiles,
       act.standard_value,
       act.standard_type
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'  -- Lọc theo loại hoạt tính
  AND act.standard_relation = '=' -- Lọc theo quan hệ
  AND act.standard_value IS NOT NULL -- Lọc giá trị NULL
  AND md.canonical_smiles IS NOT NULL
  AND act.standard_value ~ '^[0-9\.]+$' -- Lọc giá trị số
LIMIT 100;
```

Lỗi bạn gặp phải "ERROR: operator does not exist: numeric ~ unknown" xảy ra vì PostgreSQL không thể sử dụng toán tử "~" (regular expression match) trực tiếp trên cột kiểu numeric. Để khắc phục, bạn cần ép kiểu cột 'standard_value' về text trước khi so sánh:

```sql
-- Lấy 100 dòng dữ liệu từ bảng 'activities' và 'molecule_dictionary'
SELECT md.chembl_id,
       md.canonical_smiles,
       act.standard_value,
```

```
       act.standard_type
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'  -- Lọc theo loại hoạt tính
  AND act.standard_relation = '=' -- Lọc theo quan hệ
  AND act.standard_value IS NOT NULL -- Lọc giá trị NULL
  AND md.canonical_smiles IS NOT NULL
  AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$' -- Lọc giá trị số
LIMIT 100;
```

Python (để đọc dữ liệu, tính toán descriptor và xây dựng mô hình):

```
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 MinMaxScaler

# Cấu hình đường dẫn
base_path = "." # Thay đổi nếu cần
data_path = os.path.join(base_path, "data")
output_path = os.path.join(base_path, "output")

# Đọc dữ liệu từ file CSV
data = pd.read_csv(os.path.join(data_path, "chembl_data.csv"))

# Hàm tính toán các descriptor 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['HBD'] = Descriptors.NumHDonors(mol)
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
    return descriptors

# Áp dụng hàm tính toán descriptor và xử lý các giá trị bị thiếu
data['descriptors'] = data['canonical_smiles'].apply(calculate_descriptors)
data = data.dropna(subset=['descriptors'])
data = data[data['descriptors'].apply(lambda x: all(isinstance(v, (int, float)) for v
in x.values()))]

# Chuyển đổi descriptors thành các cột riêng biệt
data = pd.concat([data.drop(['descriptors'], axis=1),
data['descriptors'].apply(pd.Series)], axis=1)

# Chuyển đổi IC50 thành pIC50 (ví dụ)
data['pIC50'] = -np.log10(data['standard_value'] / 1e9)

# Chuẩn bị dữ liệu cho mô hình
X = data[['MolWt', 'LogP', 'HBD', 'HBA']]
y = data['pIC50']

# 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)

# Chuẩn hóa dữ liệu
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Xây dựng mô hình hồi quy tuyến tính
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán và đánh giá mô hình
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse}")
print(f"R^2: {r2}")

# Lưu kết quả (ví dụ)
# data.to_csv(os.path.join(output_path, "processed_data.csv"), index=False)
```

- **Giải thích code:**
  - Đọc dữ liệu từ file CSV đã tạo từ truy vấn SQL.
  - Sử dụng RDKit để tính toán các descriptor phân tử (MolWt, LogP, HBD, HBA).
  - Chuyển đổi IC50 thành pIC50.
  - Chia dữ liệu thành tập huấn luyện và tập kiểm tra.
  - Xây dựng mô hình hồi quy tuyến tính và đánh giá hiệu năng.
- **Lưu ý:**
  - Bạn có thể thay đổi các descriptor phân tử và mô hình học máy tùy theo nhu cầu.
  - Đảm bảo rằng thư viện scikit-learn của bạn là phiên bản mới nhất để hỗ trợ `squared=False` trong hàm `mean_squared_error`. Nếu không, bạn có thể bỏ tham số này (mặc định là True).

4. Ví Dụ Code (Code Examples)

Dưới đây là 5 ví dụ code SQL và Python mẫu để bạn tham khảo:

Ví dụ 1: Lọc dữ liệu theo khoảng giá trị hoạt tính (Filtering by Activity Range)

- **SQL:**

```
SELECT md.chembl_id, md.canonical_smiles, act.standard_value
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_value BETWEEN 100 AND 1000
LIMIT 100;
```

- **Python:**

```
# Sau khi đọc dữ liệu từ CSV
data_filtered = data[(data['standard_value'] >= 100) & (data['standard_value'] <=
1000)]
```

Ví dụ 2: Tính toán descriptor TPSA (Calculating TPSA Descriptor)

- **Python:**

```
from rdkit.Chem import rdMolDescriptors

def calculate_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    return rdMolDescriptors.CalcTPSA(mol)

data['TPSA'] = data['canonical_smiles'].apply(calculate_tpsa)
```

Ví dụ 3: Xây dựng mô hình Random Forest (Building a Random Forest Model)

- **Python:**

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Ví dụ 4: Trực quan hóa kết quả dự đoán (Visualizing Prediction Results)

- **Python:**

```
import matplotlib.pyplot as plt

plt.scatter(y_test, y_pred)
plt.xlabel("Actual pIC50")
plt.ylabel("Predicted pIC50")
plt.title("Actual vs. Predicted pIC50")
plt.show()
```

Ví dụ 5: Sử dụng Pipeline trong scikit-learn (Using Pipeline in scikit-learn)

- **Python:**

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LinearRegression())
])

pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
```

Lưu ý quan trọng:

- **Đảm bảo rằng bạn đã cài đặt tất cả các thư viện cần thiết (RDKit, scikit-learn, pandas, numpy).**
- **Điều chỉnh code cho phù hợp với cấu trúc dữ liệu và mục tiêu phân tích cụ thể của bạn.**
- **Thường xuyên kiểm tra và xử lý các lỗi có thể xảy ra trong quá trình thực thi.**

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