PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Liên kết ChEMBL với UniProt để tra protein

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 lưu ý quan trọng để bạn triển khai dự án một cách hiệu quả.

1. Phân tích Mô hình (Analysis of the Model)

Chúng ta sẽ tập trung vào việc khám phá dữ liệu ChEMBL 35 để tìm kiếm các hợp chất tiềm năng có hoạt tính sinh học. Dưới đây là một số bước chính trong mô hình phân tích của chúng ta:

- **Data Extraction and Preparation (Trích xuất và Chuẩn bị Dữ liệu):**
  - 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, loại bỏ các giá trị trùng lặp hoặc không hợp lệ.
  - Lưu dữ liệu đã xử lý vào các file CSV để sử dụng trong Python.
- **Feature Engineering (Xây dựng Đặc trưng):**
  - Sử dụng RDKit để tính toán các đặc trưng phân tử từ cấu trúc hóa học của các hợp chất (ví dụ: trọng lượng phân tử, logP, số lượng liên kết, v.v.).
  - Kết hợp các đặc trưng phân tử với dữ liệu hoạt tính sinh học (ví dụ: IC50, Ki) để tạo ra một tập dữ liệu hoàn chỉnh.
- **Exploratory Data Analysis (EDA) (Phân tích Thăm dò Dữ liệu):**
  - Sử dụng các kỹ thuật thống kê và trực quan hóa để khám phá dữ liệu và xác định các xu hướng, mối quan hệ và các điểm dữ liệu ngoại lệ.
  - Ví dụ: phân phối của các giá trị hoạt tính, mối tương quan giữa các đặc trưng phân tử và hoạt tính.
- **Model Building (Xây dựng Mô hình):**
  - Xây dựng các mô hình học máy để dự đoán hoạt tính sinh học của các hợp chất dựa trên các đặc trưng phân tử.
  - Sử dụng các thuật toán như hồi quy tuyến tính, random forest, support vector machines (SVM), hoặc mạng nơ-ron.
- **Model Evaluation (Đánh giá Mô hình):**
  - Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp (ví dụ: R-squared, mean squared error, root mean squared error).
  - Sử dụng kỹ thuật cross-validation để đảm bảo tính tổng quát của mô hình.

2. Hướng dẫn Song ngữ (Bilingual Guide)

Dưới đây là hướng dẫn chi tiết từng bước, kèm theo ví dụ code SQL và Python:

Bước 1: Trích xuất Dữ liệu từ ChEMBL 35 (Data Extraction from ChEMBL 35)

- **SQL:**

```sql
-- Lấy thông tin về các hợp chất và hoạt tính của chúng
SELECT
    md.molregno, -- Molecule Registry Number (Số đăng ký phân tử)
    cs.canonical_smiles, -- Canonical SMILES string (Chuỗi SMILES chuẩn)
    act.standard_type, -- Standard activity type (Loại hoạt tính chuẩn)
    act.standard_value, -- Standard activity value (Giá trị hoạt tính chuẩn)
    act.standard_units -- Standard activity units (Đơn vị hoạt tính chuẩn)
```

```
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 IC50
  AND act.standard_value IS NOT NULL  -- bỏ qua các giá trị NULL
  AND act.standard_value::TEXT ~ '^[0-9\.]+$' -- Lọc các giá trị số
LIMIT 100; -- Lấy 100 dòng dữ liệu
```

Lưu ý: Sửa lỗi `ERROR: operator does not exist: numeric ~ unknown`:

- Sử dụng `act.standard_value::TEXT ~ '^[0-9\.]+$'` để ép kiểu `standard_value` sang text trước khi so sánh với regular expression.
- **Explanation:**
    - This SQL query retrieves data from the `molecule_dictionary`, `compound_structures`, and `activities` tables in the ChEMBL database.
    - It selects the molecule registry number, canonical SMILES, standard activity type, standard activity value, and standard activity units.
    - It filters the data to include only IC50 activity values and numeric values.
    - The `LIMIT 100` clause restricts the output to the first 100 rows.
- **Giải thích:**
    - Câu truy vấn SQL này lấy dữ liệu từ các bảng `molecule_dictionary`, `compound_structures` và `activities` trong cơ sở dữ liệu ChEMBL.
    - Nó chọn số đăng ký phân tử, chuỗi SMILES chuẩn, loại hoạt tính chuẩn, giá trị hoạt tính chuẩn và đơn vị hoạt tính chuẩn.
    - Nó lọc dữ liệu để chỉ bao gồm các giá trị hoạt tính IC50 và các giá trị số.
    - Mệnh đề `LIMIT 100` giới hạn đầu ra chỉ còn 100 hàng đầu tiên.

Bước 2: Phân tích Dữ liệu bằng Python và RDKit (Data Analysis with Python and RDKit)

```
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

# Define base path
base_path = "."  # assuming the notebook is in the root directory

# Load data from CSV file
data_path = os.path.join(base_path, 'data', 'chembl_ic50_data.csv')  # Replace with
your actual file name
df = pd.read_csv(data_path)

# Preprocessing: Handle missing values and convert IC50 to pIC50
df.dropna(subset=['standard_value', 'canonical_smiles'], inplace=True)
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df.dropna(subset=['standard_value'], inplace=True)  # Drop rows where conversion
failed
```

```python
# Convert IC50 to pIC50
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Convert nM to Molar

# RDKit Feature Calculation
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    descriptors['MW'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    # You can add more RDKit descriptors here
    return descriptors

# Apply descriptor calculation
df['descriptors']  = df['canonical_smiles'].apply(lambda x: calculate_descriptors(x))
df.dropna(subset=['descriptors'], inplace=True)  # Drop rows where descriptor
calculation failed

# Convert descriptors to columns
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

# Model Building
X = df[['MW', 'LogP']].fillna(0)  # Use molecular weight and LogP as features, fill
NaN with 0
y = df['pIC50']

# 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)

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
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}")

# Save the model (optional)
#import joblib
#joblib.dump(model, 'linear_regression_model.pkl')
```

- **Explanation:**
    - This Python script loads the data from the CSV file, preprocesses it, calculates molecular descriptors using RDKit, builds a linear regression model, and evaluates the model's performance.
    - It uses `pandas` for data manipulation, `RDKit` for descriptor calculation, and `scikit-learn` for model building and evaluation.

- The `mean_squared_error` function calculates the mean squared error between the predicted and actual values. The `squared=False` parameter has been removed to maintain compatibility with older versions of scikit-learn.

- **Giải thích:**
  - Đoạn script Python này tải dữ liệu từ file CSV, tiền xử lý nó, tính toán các descriptor phân tử bằng RDKit, xây dựng mô hình hồi quy tuyến tính và đánh giá hiệu suất của mô hình.
  - Nó sử dụng `pandas` để thao tác dữ liệu, `RDKit` để tính toán descriptor và `scikit-learn` để xây dựng và đánh giá mô hình.
  - Hàm `mean_squared_error` tính toán sai số bình phương trung bình giữa các giá trị dự đoán và thực tế. Tham số `squared=False` đã được loại bỏ để duy trì khả năng tương thích với các phiên bản scikit-learn cũ hơn.

Bước 3: EDA (Exploratory Data Analysis)

```
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution of pIC50 values
plt.figure(figsize=(8, 6))
sns.histplot(df['pIC50'].dropna(), kde=True)
plt.title('Distribution of pIC50 Values')
plt.xlabel('pIC50')
plt.ylabel('Frequency')
plt.show()

# Scatter plot of MW vs LogP
plt.figure(figsize=(8, 6))
sns.scatterplot(x='MW', y='LogP', data=df)
plt.title('Molecular Weight vs LogP')
plt.xlabel('Molecular Weight')
plt.ylabel('LogP')
plt.show()
```

- **Explanation:**
  - These Python code uses `matplotlib` and `seaborn` to visualize the distribution of pIC50 values and the relationship between molecular weight and LogP.
- **Giải thích:**
  - Đoạn code Python này sử dụng `matplotlib` và `seaborn` để trực quan hóa sự phân bố của các giá trị pIC50 và mối quan hệ giữa trọng lượng phân tử và LogP.

3. Các Ví dụ Code Mẫu (Code Examples)

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

Ví dụ 1: Truy vấn SQL để lấy các hợp chất có hoạt tính cao (SQL Query to Get Highly Active Compounds)

```
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_value IS NOT NULL
  AND act.standard_value <= 100 -- Lọc các hợp chất có IC50 <= 100 nM
LIMIT 100;
```

Ví dụ 2: Tính toán số lượng vòng trong phân tử (Calculate Number of Rings in a Molecule)

```
from rdkit import Chem
from rdkit.Chem import rdMolDescriptors

def calculate_num_rings(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    return rdMolDescriptors.CalcNumRings(mol)

smiles = 'CC(=O)Oc1ccccc1C(=O)O'  # Example SMILES string (Aspirin)
num_rings = calculate_num_rings(smiles)
print(f"Number of rings: {num_rings}")
```

Ví dụ 3: Lọc các hợp chất dựa trên trọng lượng phân tử (Filter Compounds Based on Molecular Weight)

```
# Assuming df is your Pandas DataFrame
df_filtered = df[(df['MW'] >= 200) & (df['MW'] <= 500)] #Filter compounds with MW
between 200 and 500
print(df_filtered.head())
```

Ví dụ 4: Xây dựng mô hình Random Forest (Build Random Forest Model)

```
from sklearn.ensemble import RandomForestRegressor

# Model Building
X = df[['MW', 'LogP']].fillna(0)  # Use molecular weight and LogP as features, fill
NaN with 0
y = df['pIC50']

# 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)

# Train a Random Forest model
model = RandomForestRegressor(n_estimators=100, random_state=42) # Number of trees =
100
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
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}")
```

Ví dụ 5: Lưu kết quả vào file CSV (Save Results to CSV File)

```
# Assuming df is your Pandas DataFrame with calculated descriptors and predictions
df.to_csv('chembl_results.csv', index=False)
```

Lưu ý Quan trọng (Important Notes)

- **Phiên bản Scikit-learn:** Đảm bảo bạn sử dụng phiên bản scikit-learn phù hợp. Nếu bạn gặp lỗi liên quan đến `squared=False`, hãy nâng cấp scikit-learn hoặc loại bỏ tham số này.
- **Đường dẫn File:** Sử dụng `os.path.join(base_path, ...)` để đảm bảo tính di động của code trên các hệ thống khác nhau.
- **Xử lý Lỗi:** Thêm các khối `try...except` để xử lý các lỗi có thể xảy ra trong quá trình tính toán descriptor hoặc xây dựng mô hình.
- **Tài nguyên Tính toán:** Tính toán các descriptor phân tử có thể tốn nhiều thời gian. Hãy cân nhắc sử dụng các kỹ thuật tối ưu hóa hoặc tính toán song song để tăng tốc quá trình.
- **Lọc Dữ liệu:** Luôn lọc dữ liệu để loại bỏ các giá trị không hợp lệ hoặc thiếu trước khi tiến hành phân tích.
- **Regular Expression:** Việc sử dụng regular expression trong SQL có thể khác nhau tùy thuộc vào hệ quản trị cơ sở dữ liệu. Hãy kiểm tra cú pháp chính xác cho PostgreSQL.

Chúc bạn thành công với dự án phân tích dữ liệu ChEMBL 35! Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi.