PharmApp Suite

🧠 AI for Drug Discovery and Development 🍼

🧩 Topic: Dự đoán cách thức gắn ligand bằng docking

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit theo chuẩn MLOps. Dựa trên thông tin bạn cung cấp, chúng ta sẽ cùng nhau xây dựng quy trình làm việc hiệu quả và giải quyết các vấn đề phát sinh.

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

Chúng ta sẽ tập trung vào việc xây dựng một quy trình làm việc rõ ràng, dễ theo dõi và có khả năng mở rộng. Dưới đây là các bước chính:

- **Truy vấn dữ liệu từ ChEMBL 35:** Sử dụng SQL để trích xuất thông tin cần thiết từ cơ sở dữ liệu ChEMBL 35. Chúng ta sẽ tập trung vào các bảng như `activities`, `molecule_dictionary`, và `compound_structures` để thu thập dữ liệu về hoạt tính sinh học, thông tin phân tử, và cấu trúc hóa học.
- **Tiền xử lý dữ liệu:** Làm sạch và chuẩn hóa dữ liệu, xử lý các giá trị thiếu, và chuyển đổi dữ liệu về định dạng phù hợp cho phân tích.
- **Tính toán đặc trưng phân tử:** Sử dụng RDKit để tính toán các đặc trưng phân tử (molecular features) từ cấu trúc hóa học của các hợp chất. Các đặc trưng này có thể bao gồm các thuộc tính vật lý hóa học, descriptor cấu trúc, và fingerprint.
- **Phân tích thống kê và mô hình hóa:** Sử dụng các kỹ thuật thống kê và học máy để khám phá mối quan hệ giữa các đặc trưng phân tử và hoạt tính sinh học. Chúng ta có thể sử dụng các mô hình như hồi quy tuyến tính, random forest, hoặc mạng neural để dự đoán hoạt tính của các hợp chất mới.
- **Đánh giá mô hình:** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp như RMSE, R-squared, và AUC.
- **Trực quan hóa dữ liệu:** Sử dụng các công cụ trực quan hóa để khám phá dữ liệu và trình bày kết quả một cách dễ hiểu.

2. Hướng dẫn song ngữ (Bilingual Guidance)

2.1. SQL

- **Mục tiêu:** Trích xuất dữ liệu hoạt tính sinh học (bioactivity data) và thông tin phân tử (molecule information) từ cơ sở dữ liệu ChEMBL.
- **Ví dụ:**

```sql
-- English
-- Extracting activity data and molecule information for a specific target
SELECT
    act.activity_id,
    mol.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    cs.canonical_smiles
FROM
    activities act
JOIN
    molecule_dictionary mol ON act.molregno = mol.molregno
JOIN
    compound_structures cs ON mol.molregno = cs.molregno
```

```
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND act.standard_value ~ '^[0-9\.]+$' -- Ensure standard_value contains only
numbers and dots
LIMIT 100;

-- Vietnamese
-- Trích xuất dữ liệu hoạt tính và thông tin phân tử cho một mục tiêu cụ thể
SELECT
    act.activity_id,
    mol.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    cs.canonical_smiles
FROM
    activities act
JOIN
    molecule_dictionary mol ON act.molregno = mol.molregno
JOIN
    compound_structures cs ON mol.molregno = cs.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND act.standard_value ~ '^[0-9\.]+$' -- Đảm bảo standard_value chỉ chứa số và dấu
chấm
LIMIT 100;
```

- **Giải thích:**
  - Câu truy vấn này chọn các cột `activity_id`, `chembl_id`, `standard_type`, `standard_value`, `standard_units`, và `canonical_smiles` từ các bảng `activities`, `molecule_dictionary`, và `compound_structures`.
  - Nó lọc dữ liệu để chỉ bao gồm các hoạt động có `standard_type` là 'IC50' và `standard_units` là 'nM', và `standard_value` không rỗng, lớn hơn 0, và chỉ chứa số và dấu chấm.
  - `LIMIT 100` giới hạn kết quả trả về 100 dòng.
- **Sửa lỗi:**
  - Lỗi `ERROR: operator does not exist: numeric ~ unknown` xảy ra do bạn đang cố gắng sử dụng toán tử ~ (regex match) trên cột kiểu số (`standard_value`). Để khắc phục, bạn có thể ép kiểu `standard_value` sang kiểu text trước khi so sánh:

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

2.2. Python

- **Mục tiêu:** Đọc dữ liệu từ file CSV, tính toán đặc trưng phân tử bằng RDKit, và xây dựng mô hình học máy.
- **Ví dụ:**

```
# English
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```python
from sklearn.metrics import mean_squared_error, r2_score
import os
import numpy as np

# Define base path
base_path = "../data"  # Adjust if needed

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

# Function to calculate molecular weight
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return np.nan

# Apply the function to create a new column 'molecular_weight'
df['molecular_weight'] = df['canonical_smiles'].apply(calculate_mw)

# Drop rows with NaN values in 'standard_value' or 'molecular_weight'
df = df.dropna(subset=['standard_value', 'molecular_weight'])

# Prepare data for modeling
X = df[['molecular_weight']]  # Feature: Molecular Weight
y = df['standard_value']       # Target: standard_value

# 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 on the test set
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}")

# Vietnamese
import pandas as pd
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, r2_score
import os
import numpy as np

# Định nghĩa đường dẫn gốc
```

```
base_path = "../data"  # Điều chỉnh nếu cần

# Tải dữ liệu từ file CSV
csv_file = "chembl_activity_data.csv" # Thay thế bằng tên file CSV của bạn
data_path = os.path.join(base_path, csv_file)
df = pd.read_csv(data_path)

# Hàm tính toán khối lượng phân tử
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return np.nan

# Áp dụng hàm để tạo cột mới 'molecular_weight'
df['molecular_weight'] = df['canonical_smiles'].apply(calculate_mw)

# Loại bỏ các hàng có giá trị NaN trong 'standard_value' hoặc 'molecular_weight'
df = df.dropna(subset=['standard_value', 'molecular_weight'])

# Chuẩn bị dữ liệu cho mô hình hóa
X = df[['molecular_weight']]  # Đặc trưng: Khối lượng phân tử
y = df['standard_value']       # Mục tiêu: standard_value

# 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)

# Huấn luyện 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)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

- **Giải thích:**
  - Đoạn code này đọc dữ liệu từ file CSV, tính toán khối lượng phân tử từ SMILES sử dụng RDKit, và xây dựng một mô hình hồi quy tuyến tính để dự đoán `standard_value` dựa trên khối lượng phân tử.
  - Nó chia dữ liệu thành tập huấn luyện và tập kiểm tra, huấn luyện mô hình trên tập huấn luyện, và đánh giá hiệu suất của mô hình trên tập kiểm tra.
- **Sửa lỗi:**
  - Nếu bạn gặp lỗi `TypeError: mean_squared_error() got an unexpected keyword argument 'squared'`, điều này có nghĩa là phiên bản scikit-learn của bạn quá cũ. Bạn có thể nâng cấp scikit-learn bằng lệnh `pip install -U scikit-learn`. Nếu bạn không muốn nâng cấp, bạn có thể bỏ tham số `squared=False` (trong phiên bản cũ, mặc định `squared=True`, trả về MSE thay vì RMSE).

3. Ví dụ code SQL và Python mẫu (Example SQL and Python Code Snippets)

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 (Filtering data by value range)

- **SQL:**

```sql
-- English
-- Select compounds with IC50 values between 100 and 1000 nM
SELECT mol.chembl_id, act.standard_value
FROM activities act
JOIN molecule_dictionary mol ON act.molregno = mol.molregno
WHERE act.standard_type = 'IC50'
AND act.standard_units = 'nM'
AND act.standard_value >= 100
AND act.standard_value <= 1000
LIMIT 100;

-- Vietnamese
-- Chọn các hợp chất có giá trị IC50 nằm giữa 100 và 1000 nM
SELECT mol.chembl_id, act.standard_value
FROM activities act
JOIN molecule_dictionary mol ON act.molregno = mol.molregno
WHERE act.standard_type = 'IC50'
AND act.standard_units = 'nM'
AND act.standard_value >= 100
AND act.standard_value <= 1000
LIMIT 100;
```

- **Python:**

```python
# English
# Filter DataFrame for IC50 values between 100 and 1000
filtered_df = df[(df['standard_type'] == 'IC50') & (df['standard_units'] == 'nM') &
(df['standard_value'] >= 100) & (df['standard_value'] <= 1000)]

# Vietnamese
# Lọc DataFrame cho các giá trị IC50 nằm giữa 100 và 1000
filtered_df = df[(df['standard_type'] == 'IC50') & (df['standard_units'] == 'nM') &
(df['standard_value'] >= 100) & (df['standard_value'] <= 1000)]
```

Ví dụ 2: Tính toán logP sử dụng RDKit (Calculating logP using RDKit)

- **Python:**

```python
# English
from rdkit.Chem import Crippen

def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Crippen.MolLogP(mol)
    else:
        return None

df['logp'] = df['canonical_smiles'].apply(calculate_logp)

# Vietnamese
from rdkit.Chem import Crippen

def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Crippen.MolLogP(mol)
```

```
    else:
        return None

df['logp'] = df['canonical_smiles'].apply(calculate_logp)
```

Ví dụ 3: Trích xuất dữ liệu cho một protein cụ thể (Extracting data for a specific protein)

- **SQL:**

```
-- English
-- Select activity data for a specific target (e.g., with target_chembl_id
'CHEMBL205')
SELECT act.activity_id, mol.chembl_id, act.standard_value
FROM activities act
JOIN molecule_dictionary mol ON act.molregno = mol.molregno
JOIN target_dictionary td ON act.tid = td.tid
WHERE td.target_chembl_id = 'CHEMBL205'
AND act.standard_type = 'IC50'
AND act.standard_units = 'nM'
LIMIT 100;

-- Vietnamese
-- Chọn dữ liệu hoạt tính cho một mục tiêu cụ thể (ví dụ, với target_chembl_id
'CHEMBL205')
SELECT act.activity_id, mol.chembl_id, act.standard_value
FROM activities act
JOIN molecule_dictionary mol ON act.molregno = mol.molregno
JOIN target_dictionary td ON act.tid = td.tid
WHERE td.target_chembl_id = 'CHEMBL205'
AND act.standard_type = 'IC50'
AND act.standard_units = 'nM'
LIMIT 100;
```

Ví dụ 4: Sử dụng Random Forest để dự đoán hoạt tính (Using Random Forest to predict activity)

- **Python:**

```
# English
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Assuming you have features X and target y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

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

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}")

# Vietnamese
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Giả sử bạn có các đặc trưng X và mục tiêu y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

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

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}")
```

Ví dụ 5: Tính toán Tanimoto similarity giữa các phân tử (Calculating Tanimoto similarity between molecules)

- **Python:**

```
# English
from rdkit import DataStructs
from rdkit.Chem import AllChem

def calculate_tanimoto_similarity(smiles1, smiles2):
    mol1 = Chem.MolFromSmiles(smiles1)
    mol2 = Chem.MolFromSmiles(smiles2)

    if mol1 and mol2:
        fp1 = AllChem.GetMorganFingerprintAsBitVect(mol1, 2, nBits=2048)
        fp2 = AllChem.GetMorganFingerprintAsBitVect(mol2, 2, nBits=2048)
        return DataStructs.TanimotoSimilarity(fp1, fp2)
    else:
        return None

# Example usage:
smiles1 = "CC(=O)Oc1ccccc1C(=O)O" # Aspirin
smiles2 = "CC(=O)Nc1ccccc1O" # Paracetamol
similarity = calculate_tanimoto_similarity(smiles1, smiles2)
print(f"Tanimoto Similarity: {similarity}")

# Vietnamese
from rdkit import DataStructs
from rdkit.Chem import AllChem

def calculate_tanimoto_similarity(smiles1, smiles2):
    mol1 = Chem.MolFromSmiles(smiles1)
    mol2 = Chem.MolFromSmiles(smiles2)

    if mol1 and mol2:
        fp1 = AllChem.GetMorganFingerprintAsBitVect(mol1, 2, nBits=2048)
        fp2 = AllChem.GetMorganFingerprintAsBitVect(mol2, 2, nBits=2048)
        return DataStructs.TanimotoSimilarity(fp1, fp2)
    else:
        return None

# Ví dụ sử dụng:
smiles1 = "CC(=O)Oc1ccccc1C(=O)O" # Aspirin
smiles2 = "CC(=O)Nc1ccccc1O" # Paracetamol
```

```
similarity = calculate_tanimoto_similarity(smiles1, smiles2)
print(f"Tanimoto Similarity: {similarity}")
```

Lưu ý quan trọng:

- Hãy thay thế các tên file và đường dẫn phù hợp với cấu trúc thư mục của bạn.
- Điều chỉnh các tham số của mô hình học máy để đạt được hiệu suất tốt nhất.
- Sử dụng các kỹ thuật tiền xử lý dữ liệu phù hợp để làm sạch và chuẩn hóa dữ liệu.
- Thường xuyên kiểm tra và đánh giá mô hình của bạn để đảm bảo rằng nó hoạt động tốt.
- Khi gặp lỗi, hãy đọc kỹ thông báo lỗi và tìm kiếm giải pháp trên Google hoặc Stack Overflow.

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