PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Trực quan hóa họ protein từ ChEMBL

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 hiểu rõ tầm quan trọng của việc khai thác thông tin từ cơ sở dữ liệu lớn như ChEMBL để thúc đẩy quá trình phát triển thuốc.

Dựa trên thông tin bạn cung cấp, tôi sẽ giúp bạn xây dựng quy trình phân tích, khắc phục lỗi và cung cấp các ví dụ code mẫu theo chuẩn AIMLOps.

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

Mục tiêu của bạn là phân tích dữ liệu ChEMBL 35 để phục vụ nghiên cứu và phát triển thuốc. Dưới đây là một số mô hình phân tích phổ biến bạn có thể áp dụng:

- **Phân tích SAR (Structure-Activity Relationship):** Nghiên cứu mối quan hệ giữa cấu trúc hóa học của các phân tử và hoạt tính sinh học của chúng. Điều này giúp xác định các nhóm thế (functional groups) quan trọng cho hoạt tính.
- **Phân tích QSAR (Quantitative Structure-Activity Relationship):** Xây dựng các mô hình toán học để dự đoán hoạt tính sinh học dựa trên các thuộc tính hóa học tính toán được (ví dụ: trọng lượng phân tử, logP, số lượng liên kết hydro).
- **Phân tích đa dạng hóa học (Chemical Diversity Analysis):** Đánh giá sự đa dạng của các phân tử trong một tập hợp dữ liệu, giúp chọn ra các phân tử đại diện cho các vùng không gian hóa học khác nhau.
- **Phân tích sàng lọc ảo (Virtual Screening):** Sử dụng các mô hình QSAR hoặc docking để sàng lọc một lượng lớn các phân tử và xác định các ứng cử viên tiềm năng cho thử nghiệm sinh học.
- **Phân tích liên kết hóa học (Chemical Linkage Analysis):** Nghiên cứu các liên kết hóa học có trong các phân tử có hoạt tính, để tìm ra các mẫu cấu trúc chung (common structural patterns) liên quan đến hoạt tính.

2. Hướng dẫn Song ngữ (Bilingual Guidance):

Tôi sẽ cung cấp hướng dẫn bằng cả tiếng Anh và tiếng Việt để bạn dễ dàng theo dõi và thực hiện.

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

Tôi sẽ cung cấp các đoạn code SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35 và các đoạn code Python sử dụng RDKit để xử lý và phân tích dữ liệu.

4. Ví dụ Code (Code Examples):

Tôi sẽ cung cấp 5 ví dụ code SQL và Python mẫu để bạn bắt đầu.

Khắc phục lỗi (Error Handling):

- **Lỗi SQL (SQL Error):** "ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '[1]+$'"
    - **Giải thích (Explanation):** Lỗi này xảy ra do bạn đang cố gắng sử dụng toán tử `~` (regex match) trên một cột kiểu số (`numeric`). Toán tử này chỉ dùng cho kiểu dữ liệu văn bản (`text`).

[1] 0-9.

    - **Khắc phục (Solution):** Bạn cần chuyển đổi cột `standard_value` sang kiểu văn bản trước khi sử dụng toán tử ~.
    - **English:** This error occurs because you are trying to use the ~ (regex match) operator on a numeric column. This operator is only for text data types. You need to convert the `standard_value` column to text before using the ~ operator.
- **Lỗi Python (Python Error):** "phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean_squared_error"
    - **Giải thích (Explanation):** Phiên bản scikit-learn bạn đang sử dụng quá cũ và không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`.
    - **Khắc phục (Solution):** Nâng cấp phiên bản scikit-learn của bạn lên phiên bản mới nhất bằng lệnh `pip install --upgrade scikit-learn`. Nếu không thể nâng cấp, bạn có thể tính căn bậc hai của MSE (Mean Squared Error) để có được RMSE (Root Mean Squared Error).
    - **English:** The version of scikit-learn you are using is too old and does not support the `squared=False` parameter in the `mean_squared_error` function. Upgrade your scikit-learn version to the latest version using `pip install --upgrade scikit-learn`. If you cannot upgrade, you can calculate the square root of MSE to get RMSE.

Ví dụ Code (Code Examples):

Dưới đây là 5 ví dụ code SQL và Python mẫu, tập trung vào phân tích SAR và QSAR:

Ví dụ 1: Lấy dữ liệu hoạt tính và cấu trúc (Fetching Activity and Structure Data)

- **SQL:**

```sql
-- Lấy 100 dòng dữ liệu hoạt tính kháng một target cụ thể (ví dụ: CHEMBL240)
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    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.assay_id IN (SELECT assay_id FROM assays WHERE target_chembl_id = 'CHEMBL240')
    AND 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;
```

- **English:**

```sql
-- Fetch 100 rows of activity data against a specific target (e.g., CHEMBL240)
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    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.assay_id IN (SELECT assay_id FROM assays WHERE target_chembl_id = 'CHEMBL240')
    AND 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:**

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

base_path = "." # Thay đổi nếu cần

# Đường dẫn đến file CSV chứa dữ liệu lấy từ SQL
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")

# Đọc dữ liệu từ file CSV
df = pd.read_csv(csv_file_path)

# Hàm tính toán các thuộc tính hóa học sử dụng RDKit
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        descriptors = {desc_name: desc_func(mol) for desc_name, desc_func in
Descriptors.descList}
        return descriptors
    else:
        return None

# Áp dụng hàm tính toán thuộc tính cho cột 'canonical_smiles'
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# In ra một vài dòng đầu của DataFrame
print(df.head())
```

- **English:**

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

base_path = "." # Change if needed

# Path to the CSV file containing data fetched from SQL
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")

# Read data from CSV file
df = pd.read_csv(csv_file_path)

# Function to calculate chemical descriptors using RDKit
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        descriptors = {desc_name: desc_func(mol) for desc_name, desc_func in
```

```
Descriptors.descList}
        return descriptors
    else:
        return None

# Apply the descriptor calculation function to the 'canonical_smiles' column
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Print the first few rows of the DataFrame
print(df.head())
```

Ví dụ 2: Phân tích SAR đơn giản (Simple SAR Analysis)

- **SQL:** (Không cần thiết cho ví dụ này, dữ liệu đã có từ ví dụ 1)
- **Python:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Draw
import os

base_path = "." # Thay đổi nếu cần
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")
df = pd.read_csv(csv_file_path)

# Chuyển đổi cột standard_value thành số
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')

# Lọc các phân tử có hoạt tính cao (IC50 < 100 nM)
active_molecules = df[df['standard_value'] < 100]

# In ra số lượng phân tử có hoạt tính cao
print(f"Number of active molecules: {len(active_molecules)}")

# Vẽ một vài phân tử có hoạt tính cao
molecules = [Chem.MolFromSmiles(smiles) for smiles in
active_molecules['canonical_smiles'].head(4)]
img = Draw.MolsToGridImage(molecules, molsPerRow=4, subImgSize=(200,200))
img.save(os.path.join(base_path, "data", "active_molecules.png"))
```

- **English:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Draw
import os

base_path = "." # Change if needed
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")
df = pd.read_csv(csv_file_path)

# Convert the standard_value column to numeric
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')

# Filter for highly active molecules (IC50 < 100 nM)
active_molecules = df[df['standard_value'] < 100]

# Print the number of active molecules
print(f"Number of active molecules: {len(active_molecules)}")

# Draw a few highly active molecules
```

```
molecules = [Chem.MolFromSmiles(smiles) for smiles in
active_molecules['canonical_smiles'].head(4)]
img = Draw.MolsToGridImage(molecules, molsPerRow=4, subImgSize=(200,200))
img.save(os.path.join(base_path, "data", "active_molecules.png"))
```

Ví dụ 3: Tính toán LogP và TPSA (Calculating LogP and TPSA)

- **SQL:** (Không cần thiết cho ví dụ này, dữ liệu đã có từ ví dụ 1)
- **Python:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem import Crippen
from rdkit.Chem import rdMolDescriptors
import os

base_path = "." # Thay đổi nếu cần
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")
df = pd.read_csv(csv_file_path)

# Hàm tính toán LogP và TPSA
def calculate_logp_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        logp = Crippen.MolLogP(mol)
        tpsa = rdMolDescriptors.CalcTPSA(mol)
        return logp, tpsa
    else:
        return None, None

# Áp dụng hàm tính toán LogP và TPSA
df[['logp', 'tpsa']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_logp_tpsa(x)))

# In ra một vài dòng đầu của DataFrame
print(df[['canonical_smiles', 'logp', 'tpsa']].head())
```

- **English:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem import Crippen
from rdkit.Chem import rdMolDescriptors
import os

base_path = "." # Change if needed
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")
df = pd.read_csv(csv_file_path)

# Function to calculate LogP and TPSA
def calculate_logp_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        logp = Crippen.MolLogP(mol)
        tpsa = rdMolDescriptors.CalcTPSA(mol)
        return logp, tpsa
    else:
        return None, None

# Apply the LogP and TPSA calculation function
```

```
df[['logp', 'tpsa']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_logp_tpsa(x)))

# Print the first few rows of the DataFrame
print(df[['canonical_smiles', 'logp', 'tpsa']].head())
```

Ví dụ 4: Xây dựng mô hình QSAR đơn giản (Building a Simple QSAR Model)

- **SQL:** (Không cần thiết cho ví dụ này, dữ liệu đã có từ ví dụ 1 và 3)
- **Python:**

```
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
import numpy as np
import os

base_path = "." # Thay đổi nếu cần
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")
df = pd.read_csv(csv_file_path)

# Tính toán LogP và TPSA (nếu chưa có)
def calculate_logp_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        logp = Crippen.MolLogP(mol)
        tpsa = rdMolDescriptors.CalcTPSA(mol)
        return logp, tpsa
    else:
        return None, None

df[['logp', 'tpsa']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_logp_tpsa(x)))

# Chuyển đổi IC50 thành pIC50
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Chuyển đổi nM thành M

# Chọn các thuộc tính (features) và biến mục tiêu (target)
X = df[['logp', 'tpsa']].fillna(0)  # Điền giá trị NaN bằng 0
y = df['pIC50'].fillna(0)

# 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)

# Xây dựng mô hình hồi quy tuyến tính
model = LinearRegression()

# Huấn luyện mô hình
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)
```

```
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error: {rmse}")
```

- **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
from sklearn.metrics import mean_squared_error
import numpy as np
import os

base_path = "." # Change if needed
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")
df = pd.read_csv(csv_file_path)

# Calculate LogP and TPSA (if not already calculated)
def calculate_logp_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        logp = Crippen.MolLogP(mol)
        tpsa = rdMolDescriptors.CalcTPSA(mol)
        return logp, tpsa
    else:
        return None, None

df[['logp', 'tpsa']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_logp_tpsa(x)))

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

# Select features and target variable
X = df[['logp', 'tpsa']].fillna(0)  # Fill NaN values with 0
y = df['pIC50'].fillna(0)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Build a linear regression model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Predict 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}")
```

Ví dụ 5: Phân tích cấu trúc con (Substructure Analysis)

- **SQL:** (Không cần thiết cho ví dụ này, dữ liệu đã có từ ví dụ 1)
- **Python:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem import Draw
from rdkit.Chem.Draw import IPythonConsole
import os

base_path = "." # Thay đổi nếu cần
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")
df = pd.read_csv(csv_file_path)

# Định nghĩa cấu trúc con (substructure) quan tâm (ví dụ: vòng benzen)
substructure = Chem.MolFromSmiles('c1ccccc1')

# Hàm kiểm tra xem một phân tử có chứa cấu trúc con hay không
def has_substructure(smiles, substructure):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        return mol.HasSubstructMatch(substructure)
    else:
        return False

# Tạo cột mới để chỉ ra sự hiện diện của cấu trúc con
df['has_benzene'] = df['canonical_smiles'].apply(lambda x: has_substructure(x,
substructure))

# Lọc các phân tử có chứa vòng benzen
benzene_molecules = df[df['has_benzene'] == True]

# In ra số lượng phân tử có chứa vòng benzen
print(f"Number of molecules with benzene ring: {len(benzene_molecules)}")

# Vẽ một vài phân tử có chứa vòng benzen
molecules = [Chem.MolFromSmiles(smiles) for smiles in
benzene_molecules['canonical_smiles'].head(4)]
img = Draw.MolsToGridImage(molecules, molsPerRow=4, subImgSize=(200,200))
img.save(os.path.join(base_path, "data", "benzene_molecules.png"))
```

- **English:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem import Draw
from rdkit.Chem.Draw import IPythonConsole
import os

base_path = "." # Change if needed
csv_file_path = os.path.join(base_path, "data", "chembl_activity_data.csv")
df = pd.read_csv(csv_file_path)

# Define the substructure of interest (e.g., benzene ring)
substructure = Chem.MolFromSmiles('c1ccccc1')

# Function to check if a molecule contains the substructure
def has_substructure(smiles, substructure):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        return mol.HasSubstructMatch(substructure)
    else:
```

```
        return False

# Create a new column to indicate the presence of the substructure
df['has_benzene'] = df['canonical_smiles'].apply(lambda x: has_substructure(x,
substructure))

# Filter for molecules containing the benzene ring
benzene_molecules = df[df['has_benzene'] == True]

# Print the number of molecules containing the benzene ring
print(f"Number of molecules with benzene ring: {len(benzene_molecules)}")

# Draw a few molecules containing the benzene ring
molecules = [Chem.MolFromSmiles(smiles) for smiles in
benzene_molecules['canonical_smiles'].head(4)]
img = Draw.MolsToGridImage(molecules, molsPerRow=4, subImgSize=(200,200))
img.save(os.path.join(base_path, "data", "benzene_molecules.png"))
```

Lưu ý quan trọng:

- **Đường dẫn:** Hãy chắc chắn rằng bạn đã thay đổi `base_path` để phù hợp với cấu trúc thư mục của bạn.
- **Cài đặt thư viện:** Đảm bảo bạn đã cài đặt tất cả các thư viện cần thiết (pandas, rdkit, scikit-learn).
- **Dữ liệu:** Các ví dụ này giả định rằng bạn đã tải dữ liệu từ ChEMBL và lưu nó vào file CSV. Bạn có thể cần điều chỉnh code để phù hợp với định dạng dữ liệu của bạn.
- **Phân tích sâu hơn:** Đây chỉ là những ví dụ cơ bản. Bạn có thể mở rộng chúng bằng cách sử dụng các thuộc tính hóa học khác, các mô hình máy học phức tạp hơn và các kỹ thuật phân tích dữ liệu khác.

Hãy cho tôi biết nếu bạn có bất kỳ câu hỏi nào hoặc cần hỗ trợ thêm. Chúc bạn thành công với dự án của mình!