PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Khám phá thuốc hiếm từ ChEMBL

Tuyệt vời! Tôi rất vui được tham gia dự án thú vị này với vai trò Chuyên gia R&D Dược học. Dựa trên thông tin bạn cung cấp, tôi sẽ giúp bạn phân tích, xây dựng mô hình, cung cấp code mẫu SQL và Python (kết hợp RDKit) để làm việc với dữ liệu ChEMBL 35. Chúng ta sẽ tập trung vào việc giải quyết các lỗi bạn gặp phải và tuân thủ theo cấu trúc thư mục AIMLOps.

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

Dựa trên mã chủ đề "Topic_CheMBL_35_59", tôi hiểu rằng bạn đang muốn thực hiện một phân tích cụ thể trên dữ liệu ChEMBL 35. Để đưa ra mô hình phân tích phù hợp, tôi cần biết rõ hơn về mục tiêu nghiên cứu của bạn. Tuy nhiên, dựa trên thông tin hiện có, tôi có thể đề xuất một số hướng tiếp cận phổ biến và hữu ích trong R&D Dược học:

- **Phân tích mối tương quan Cấu trúc-Hoạt tính (SAR/QSAR):** Đây là một kỹ thuật quan trọng để xác định mối liên hệ giữa cấu trúc hóa học của một phân tử và hoạt tính sinh học của nó. Bạn có thể sử dụng RDKit để tính toán các descriptor phân tử (ví dụ: trọng lượng phân tử, logP, số lượng liên kết hydro) và sau đó sử dụng các mô hình học máy (ví dụ: hồi quy tuyến tính, Support Vector Machines, Random Forests) để dự đoán hoạt tính dựa trên các descriptor này.
- **Phân tích cụm (Clustering):** Phân tích cụm giúp bạn nhóm các phân tử có cấu trúc hoặc hoạt tính tương tự lại với nhau. Điều này có thể giúp bạn xác định các "scaffold" (khung cấu trúc) tiềm năng cho việc phát triển thuốc hoặc tìm kiếm các phân tử có hoạt tính mong muốn trong một tập dữ liệu lớn.
- **Phân tích đa dạng hóa hợp chất (Compound Diversity Analysis):** Đánh giá mức độ đa dạng của một tập hợp các phân tử. Điều này quan trọng để đảm bảo rằng bạn đang nghiên cứu một phạm vi rộng các cấu trúc hóa học và không bỏ lỡ các "hit" tiềm năng.
- **Phân tích sàng lọc ảo (Virtual Screening):** Sử dụng các mô hình tính toán để dự đoán khả năng một phân tử liên kết với một đích sinh học cụ thể (ví dụ: một protein). Điều này giúp bạn thu hẹp phạm vi các phân tử cần được thử nghiệm trong phòng thí nghiệm.

Ví dụ:

Giả sử bạn muốn xây dựng một mô hình QSAR để dự đoán hoạt tính ức chế enzyme (ví dụ: IC50) dựa trên cấu trúc phân tử. Mô hình phân tích sẽ bao gồm các bước sau:

1. **Chuẩn bị dữ liệu:** Tải dữ liệu từ cơ sở dữ liệu ChEMBL, làm sạch và chuẩn hóa dữ liệu (ví dụ: loại bỏ các giá trị trùng lặp, xử lý các giá trị thiếu).
2. **Tính toán descriptor phân tử:** Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc SMILES của các phân tử.
3. **Lựa chọn descriptor:** Chọn các descriptor phù hợp nhất để đưa vào mô hình.
4. **Xây dựng mô hình:** Sử dụng một thuật toán học máy để xây dựng mô hình dự đoán hoạt tính.
5. **Đánh giá mô hình:** Đánh giá hiệu suất của mô hình trên một tập dữ liệu kiểm tra độc lập.

2. Hướng dẫn song ngữ (Bilingual Instructions):

Dưới đây là hướng dẫn song ngữ cho các bước thực hiện:

English:

1. **Connect to the ChEMBL 35 database using psql.**
2. **Write SQL queries to extract the necessary data (e.g., compound structures, activity values).**

3. **Save the extracted data to CSV files using pgAdmin.**
4. **Load the CSV files into a Jupyter Notebook.**
5. **Use RDKit to calculate molecular descriptors from the compound structures.**
6. **Build and evaluate machine learning models using scikit-learn to predict activity.**
7. **Address the errors encountered (e.g., SQL operator error, scikit-learn version issue).**

Tiếng Việt:

1. **Kết nối đến cơ sở dữ liệu ChEMBL 35 bằng psql.**
2. **Viết các truy vấn SQL để trích xuất dữ liệu cần thiết (ví dụ: cấu trúc hợp chất, giá trị hoạt tính).**
3. **Lưu dữ liệu đã trích xuất vào các tệp CSV bằng pgAdmin.**
4. **Tải các tệp CSV vào Jupyter Notebook.**
5. **Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc của các hợp chất.**
6. **Xây dựng và đánh giá các mô hình học máy bằng scikit-learn để dự đoán hoạt tính.**
7. **Giải quyết các lỗi gặp phải (ví dụ: lỗi toán tử SQL, vấn đề phiên bản scikit-learn).**

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

a. SQL Code (English):

```sql
-- Select 100 rows of data from the activities table where the standard_type is IC50
and standard_value is numeric
SELECT
    act.molregno,
    cmp.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    cmp.structure
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
WHERE
    act.standard_type = 'IC50'
    --AND act.standard_value IS NOT NULL  -- Ensure standard_value is not null
    AND act.standard_value::text ~ '^[0-9\.]+$'  -- Check if standard_value is numeric
using regular expression
LIMIT 100;
```

SQL Code (Tiếng Việt):

```sql
-- Chọn 100 dòng dữ liệu từ bảng activities nơi standard_type là IC50 và
standard_value là số
SELECT
    act.molregno,
    cmp.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    cmp.structure
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
WHERE
    act.standard_type = 'IC50'
    --AND act.standard_value IS NOT NULL  -- Đảm bảo standard_value không phải là null
```

```
    AND act.standard_value::text ~ '^[0-9\.]+$'  -- Kiểm tra xem standard_value có
phải là số bằng cách sử dụng biểu thức chính quy
LIMIT 100;
```

Lưu ý về lỗi SQL:

Lỗi "operator does not exist: numeric ~ unknown" xảy ra vì bạn đang cố gắng sử dụng toán tử ~ (biểu thức chính quy) trên một cột kiểu số (`numeric`). Để khắc phục, bạn cần chuyển đổi cột `standard_value` thành kiểu text trước khi so sánh với biểu thức chính quy. Tôi đã sửa đổi truy vấn SQL để sử dụng `act.standard_value::text ~ '^[0-9\.]+$'` để chuyển đổi giá trị số thành text trước khi so sánh.

b. Python Code (English):

```
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

# Define the base path
base_path = "../data"  # Adjust this path as needed

# Load the CSV file
csv_file_path = os.path.join(base_path, "chembl_ic50_data.csv")  # Replace with your
actual CSV file name
df = pd.read_csv(csv_file_path)

# Data Cleaning and Preprocessing
df.dropna(subset=['standard_value', 'structure'], inplace=True)
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df.dropna(subset=['standard_value'], inplace=True)

# RDKit Molecular Descriptor Calculation
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

df['descriptors'] = df['structure'].apply(calculate_descriptors)
df.dropna(subset=['descriptors'], inplace=True)

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

# Data Splitting and Scaling
X = df[['MolWt', 'LogP', 'HBD', 'HBA']]
y = df['standard_value']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# Model Training
model = LinearRegression()
model.fit(X_train, y_train)

# Model Evaluation
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}")
```

Python Code (Tiếng Việt):

```
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

# Định nghĩa đường dẫn cơ sở
base_path = "../data"  # Điều chỉnh đường dẫn này nếu cần

# Tải tệp CSV
csv_file_path = os.path.join(base_path, "chembl_ic50_data.csv")  # Thay thế bằng tên
tệp CSV thực tế của bạn
df = pd.read_csv(csv_file_path)

# Làm sạch và tiền xử lý dữ liệu
df.dropna(subset=['standard_value', 'structure'], inplace=True)
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df.dropna(subset=['standard_value'], inplace=True)


# Tính toán descriptor phân tử bằng RDKit
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

df['descriptors'] = df['structure'].apply(calculate_descriptors)
```

```
df.dropna(subset=['descriptors'], inplace=True)

# Chuyển đổi descriptors thành các cột
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)
df.dropna(inplace=True)

# Chia dữ liệu và масштабирование
X = df[['MolWt', 'LogP', 'HBD', 'HBA']]
y = df['standard_value']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Huấn luyện mô hình
model = LinearRegression()
model.fit(X_train, y_train)

# 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}")
```

Lưu ý về lỗi Python:

- **Phiên bản scikit-learn cũ:** 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 bằng lệnh `pip install --upgrade scikit-learn`. Trong code trên, tôi đã loại bỏ tham số `squared=False` vì nó không còn cần thiết trong các phiên bản scikit-learn mới hơn.
- **Đường dẫn:** Đảm bảo rằng bạn đã thay đổi đường dẫn `base_path` và `csv_file_path` cho phù hợp với cấu trúc thư mục của bạn.

4. Ví dụ code SQL và Python (SQL and Python Code Examples):

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

Ví dụ 1: Lọc các hợp chất có trọng lượng phân tử lớn hơn 500 (Filter compounds with molecular weight > 500):

- **SQL:**

```
SELECT
    cmp.chembl_id,
    cmp.structure
FROM
    molecule_dictionary cmp
WHERE
    cmp.mol_weight > 500
LIMIT 10;
```

- **Python:**

```
from rdkit import Chem
from rdkit.Chem import Descriptors
```

```
def filter_by_molwt(smiles, threshold=500):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return False
    mol_wt = Descriptors.MolWt(mol)
    return mol_wt > threshold

df['heavy_molwt'] = df['structure'].apply(filter_by_molwt)
filtered_df = df[df['heavy_molwt'] == True]
print(filtered_df.head())
```

Ví dụ 2: Tính logP cho các hợp chất (Calculate LogP for compounds):

- **SQL:** (Không thể tính logP trực tiếp bằng SQL, cần sử dụng Python)
- **Python:**

```
from rdkit import Chem
from rdkit.Chem import Descriptors

def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    return Descriptors.MolLogP(mol)

df['logp'] = df['structure'].apply(calculate_logp)
print(df[['chembl_id', 'logp']].head())
```

Ví dụ 3: Tìm các hợp chất có số lượng liên kết hydro cho lớn hơn 5 (Find compounds with more than 5 hydrogen bond donors):

- **SQL:** (Không thể tính số lượng liên kết hydro trực tiếp bằng SQL, cần sử dụng Python)
- **Python:**

```
from rdkit import Chem
from rdkit.Chem import Descriptors

def count_hbd(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    return Descriptors.NumHDonors(mol)

df['hbd_count'] = df['structure'].apply(count_hbd)
filtered_df = df[df['hbd_count'] > 5]
print(filtered_df[['chembl_id', 'hbd_count']].head())
```

Ví dụ 4: Phân tích tần suất của các khung cấu trúc (scaffolds) (Analyze the frequency of structural frameworks (scaffolds)):

- **SQL:** (Cần kết hợp với Python để tạo scaffold)
- **Python:**

```
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import MurckoFragment
from collections import Counter

def get_murcko_scaffold(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        core = MurckoFragment.GetMurckoFragment(mol)
```

```
        return Chem.MolToSmiles(core)
    else:
        return None

df['murcko_scaffold'] = df['structure'].apply(get_murcko_scaffold)
scaffold_counts = Counter(df['murcko_scaffold'].dropna())

print("Most common Murcko scaffolds:")
for scaffold, count in scaffold_counts.most_common(10):
    print(f"{scaffold}: {count}")
```

Ví dụ 5: Tìm kiếm các cấu trúc con (Substructure search):

- **SQL:** (Không thể tìm kiếm cấu trúc con trực tiếp bằng SQL, cần sử dụng Python)
- **Python:**

```
from rdkit import Chem
from rdkit.Chem import Descriptors

# Define the substructure SMILES
substructure_smiles = 'c1ccccc1'  # Example: Benzene ring
substructure = Chem.MolFromSmiles(substructure_smiles)

def has_substructure(smiles, substructure):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return False
    return mol.HasSubstructMatch(substructure)

df['has_benzene'] = df['structure'].apply(lambda x: has_substructure(x, substructure))
benzene_containing_df = df[df['has_benzene'] == True]

print(f"Number of compounds containing benzene ring: {len(benzene_containing_df)}")
print(benzene_containing_df['structure'].head())
```

Quan trọng:

- Hãy thay thế các ví dụ này bằng các truy vấn và phân tích phù hợp với mục tiêu nghiên cứu cụ thể của bạn.
- Luôn kiểm tra và làm sạch dữ liệu của bạn trước khi thực hiện bất kỳ phân tích nào.
- Sử dụng các công cụ gỡ lỗi và kiểm tra để đảm bảo rằng code của bạn hoạt động chính xác.
- Ghi lại tất cả các bước của quy trình phân tích của bạn để đảm bảo tính tái lập.

Tôi hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án phân tích dữ liệu ChEMBL 35 của mình. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi! Chúc bạn thành công!