PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Lọc các phân tử tuân thủ quy tắc Lipinski

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 dữ liệu ChEMBL để tìm kiếm các ứng cử viên thuốc tiềm năng. Chúng ta sẽ cùng nhau xây dựng quy trình làm việc hiệu quả, tuân thủ theo chuẩn AIMLOps Template mà bạn đã đề ra.

Tổng quan về dự án

Dự án này tập trung vào việc phân tích dữ liệu từ ChEMBL 35, một cơ sở dữ liệu lớn chứa thông tin về các phân tử hoạt tính sinh học. Chúng ta sẽ sử dụng PostgreSQL để truy vấn dữ liệu, RDKit để xử lý và tính toán các đặc trưng phân tử, và Jupyter Notebook để viết code phân tích và trực quan hóa kết quả.

Các bước thực hiện

1. **Truy vấn dữ liệu từ ChEMBL 35 bằng SQL và lưu vào file CSV:** Sử dụng pgAdmin để kết nối đến cơ sở dữ liệu ChEMBL 35 và viết các câu lệnh SQL để trích xuất dữ liệu cần thiết. Lưu kết quả vào các file CSV trong thư mục `../data/`.
2. **Xử lý dữ liệu và tính toán đặc trưng phân tử bằng RDKit:** Sử dụng Jupyter Notebook để đọc dữ liệu từ các file CSV, tiền xử lý dữ liệu, tính toán các đặc trưng phân tử bằng RDKit, và lưu kết quả vào các file mới.
3. **Phân tích dữ liệu và xây dựng mô hình:** Sử dụng các thư viện Python như scikit-learn để phân tích dữ liệu, xây dựng mô hình dự đoán hoạt tính sinh học, và đánh giá hiệu năng của mô hình.
4. **Trực quan hóa kết quả:** Sử dụng các thư viện Python như matplotlib và seaborn để trực quan hóa kết quả phân tích, giúp chúng ta hiểu rõ hơn về dữ liệu và mô hình.

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

Chủ đề "Topic_CheMBL_35_4" có thể tập trung vào một số khía cạnh sau của dữ liệu ChEMBL 35:

- **Phân tích mối quan hệ cấu trúc-hoạt tính (SAR):** Tìm kiếm mối liên hệ giữa cấu trúc hóa học của các phân tử và hoạt tính sinh học của chúng. Điều này có thể giúp chúng ta xác định các nhóm chức quan trọng cho hoạt tính, và từ đó thiết kế các phân tử mới có hoạt tính tốt hơn.
- **Xây dựng mô hình dự đoán hoạt tính (Predictive modeling):** Sử dụng các thuật toán học máy để xây dựng mô hình dự đoán hoạt tính sinh học của các phân tử dựa trên cấu trúc của chúng. Mô hình này có thể được sử dụng để sàng lọc ảo các thư viện phân tử lớn, giúp chúng ta tìm kiếm các ứng cử viên thuốc tiềm năng một cách nhanh chóng và hiệu quả.
- **Phân tích đa dạng hóa học (Chemical diversity analysis):** Đánh giá sự đa dạng của các phân tử trong một tập dữ liệu. Điều này có thể giúp chúng ta đảm bảo rằng chúng ta đang khám phá không gian hóa học một cách toàn diện, và không bỏ lỡ các ứng cử viên thuốc tiềm năng.

2. Hướng dẫn song ngữ (Bilingual Instructions)

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

Bước 1: Truy vấn dữ liệu từ ChEMBL 35 bằng SQL

- **English:** Connect to the ChEMBL 35 database using pgAdmin and write SQL queries to extract the necessary data. Save the results to CSV files in the `../data/` directory.
- **Tiếng Việt:** Kết nối đến cơ sở dữ liệu ChEMBL 35 bằng pgAdmin và viết các câu lệnh SQL để trích xuất dữ liệu cần thiết. Lưu kết quả vào các file CSV trong thư mục `../data/`.

Bước 2: Xử lý dữ liệu và tính toán đặc trưng phân tử bằng RDKit

- **English:** Use Jupyter Notebook to read data from CSV files, preprocess data, calculate molecular features using RDKit, and save the results to new files.
- **Tiếng Việt:** Sử dụng Jupyter Notebook để đọc dữ liệu từ các file CSV, tiền xử lý dữ liệu, tính toán các đặc trưng phân tử bằng RDKit, và lưu kết quả vào các file mới.

Bước 3: Phân tích dữ liệu và xây dựng mô hình

- **English:** Use Python libraries like scikit-learn to analyze data, build predictive models for biological activity, and evaluate model performance.
- **Tiếng Việt:** Sử dụng các thư viện Python như scikit-learn để phân tích dữ liệu, xây dựng mô hình dự đoán hoạt tính sinh học, và đánh giá hiệu năng của mô hình.

Bước 4: Trực quan hóa kết quả

- **English:** Use Python libraries like matplotlib and seaborn to visualize the analysis results, helping us better understand the data and models.
- **Tiếng Việt:** Sử dụng các thư viện Python như matplotlib và seaborn để trực quan hóa kết quả phân tích, giúp chúng ta hiểu rõ hơn về dữ liệu và mô hình.

3. Code SQL, Python (English)

3.1. SQL Code Example

```sql
-- SQL query to extract data for compounds with IC50 values against a specific target
-- This example targets CHEMBL205 (Dopamine D4 receptor)
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    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
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.chembl_id = 'CHEMBL205'  -- Dopamine D4 receptor
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0  -- Avoid zero values
    AND act.standard_value < 10000 -- Limit to 10000 nM for activity
LIMIT 100; -- Limit to 100 rows
```

Lưu ý về lỗi SQL (Note about SQL error):

Lỗi `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'` xảy ra do bạn đang cố gắng sử dụng toán tử `~` (regular expression match) trên một cột kiểu số (numeric). Để khắc phục, bạn có thể sử dụng hàm `CAST` để chuyển đổi cột `standard_value` sang kiểu text trước khi so sánh với regular expression, hoặc bỏ điều kiện này nếu không cần thiết.

Ví dụ:

```sql
-- Remove the line causing the error:
-- AND act.standard_value ~ '^[0-9\.]+$'
```

```
-- Or cast the numeric value to text:
-- AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'
```

3.2. Python Code Example

```
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 = ".."  # Assuming the notebook is in a subdirectory

# Define the path to the CSV file
csv_file_path = os.path.join(base_path, "data", "chembl_205_ic50.csv")  # Replace with
your actual filename

# Load the data from the CSV file
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
    exit()

# Handle missing values: Remove rows with missing values in 'standard_value'
df = df.dropna(subset=['standard_value', 'canonical_smiles'])
df = df.head(100)


# RDKit function to calculate molecular descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None  # Handle invalid SMILES strings
    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    descriptors['HBD'] = Descriptors.NumHDonors(mol)
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
    return descriptors

# Apply descriptor calculation to each molecule
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Filter out rows where descriptor calculation failed (invalid SMILES)
df = df[df['descriptors'].notna()]

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

# Prepare data for machine learning
X = df[['MolWt', 'LogP', 'HBD', 'HBA']]  # Feature matrix
y = -np.log10(df['standard_value'])  # Target variable (transformed IC50)
```

```python
# Data scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, 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}")
```

Lưu ý về lỗi Python (Note about Python error):

Nếu bạn gặp lỗi về `squared=False` trong hàm `mean_squared_error`, hãy kiểm tra phiên bản scikit-learn của bạn. Nếu phiên bản cũ, hãy nâng cấp lên phiên bản mới nhất bằng lệnh:

```
pip install -U scikit-learn
```

Nếu bạn không muốn nâng cấp, bạn có thể tính căn bậc hai của MSE để có RMSE (Root Mean Squared Error), tương đương với MSE khi `squared=False`.

4. Ví dụ (Examples)

Dưới đây là 5 ví dụ về các câu lệnh SQL và Python mà bạn có thể sử dụng trong dự án của mình:

4.1. Ví dụ 1: Trích xuất dữ liệu cơ bản (Basic data extraction)

- **SQL:**

```sql
SELECT chembl_id, pref_name FROM molecule_dictionary LIMIT 10;
```

- **Python:**

```python
import pandas as pd
# Assuming you have a CSV file named 'molecule_dictionary.csv'
df = pd.read_csv('../data/molecule_dictionary.csv', nrows=10)
print(df.head())
```

4.2. Ví dụ 2: Tính toán đặc trưng phân tử đơn giản (Simple descriptor calculation)

- **Python:**

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

smiles = 'CC(=O)Oc1ccccc1C(=O)O' # Aspirin
mol = Chem.MolFromSmiles(smiles)
mol_wt = Descriptors.MolWt(mol)
print(f"Molecular weight of Aspirin: {mol_wt}")
```

4.3. Ví dụ 3: Lọc dữ liệu theo hoạt tính (Filtering data by activity)

- **SQL:**

```sql
SELECT md.chembl_id, act.standard_value
FROM molecule_dictionary md
JOIN activities act ON md.molregno = act.molregno
WHERE act.standard_type = 'IC50' AND act.standard_value < 100;
```

- **Python:**

```python
# Assuming you have a DataFrame named 'activities_df'
activities_df = pd.read_csv('../data/activities.csv')
filtered_df = activities_df[(activities_df['standard_type'] == 'IC50') &
(activities_df['standard_value'] < 100)]
print(filtered_df.head())
```

4.4. Ví dụ 4: Gom nhóm dữ liệu (Grouping data)

- **SQL:**

```sql
SELECT target_dictionary.chembl_id, COUNT(*)
FROM activities
JOIN target_dictionary ON activities.tid = target_dictionary.tid
GROUP BY target_dictionary.chembl_id
ORDER BY COUNT(*) DESC
LIMIT 10;
```

- **Python:**

```python
# Assuming you have a DataFrame named 'activities_df' and 'target_dictionary_df'
activities_df = pd.read_csv('../data/activities.csv')
target_dictionary_df = pd.read_csv('../data/target_dictionary.csv')

merged_df = pd.merge(activities_df, target_dictionary_df, on='tid')
grouped_df = merged_df.groupby('chembl_id')
['molregno'].count().sort_values(ascending=False).head(10)
print(grouped_df)
```

4.5. Ví dụ 5: Xây dựng mô hình đơn giản (Simple model building)

- **Python:**

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import numpy as np

# Assuming you have features X and target variable y
X = np.random.rand(100, 5)
y = np.random.rand(100)

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

model = LinearRegression()
model.fit(X_train, y_train)

score = model.score(X_test, y_test)
print(f"Model R^2 score: {score}")
```

Lưu ý quan trọng:

- Hãy thay đổi đường dẫn file CSV và các tham số khác cho phù hợp với dự án của bạn.
- Đảm bảo rằng bạn đã cài đặt tất cả các thư viện cần thiết (RDKit, pandas, scikit-learn, matplotlib, seaborn).
- Hãy chia nhỏ các bước phân tích và kiểm tra kết quả thường xuyên để đảm bảo rằng mọi thứ hoạt động chính xác.

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