PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Dòng thời gian về ngày phê duyệt thuốc

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 bằng RDKit. Với vai trò là một chuyên gia R&D dược học, tôi hiểu rõ tầm quan trọng của việc khai thác dữ liệu lớn để đẩy nhanh quá trình phát triển thuốc. Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu SQL và Python để bạn tham khảo:

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

Dựa trên thông tin bạn cung cấp, tôi hình dung quy trình phân tích dữ liệu của bạn như sau:

- **Giai đoạn 1: Trích xuất dữ liệu từ ChEMBL 35 (Data Extraction from ChEMBL 35)**
    - Sử dụng PostgreSQL (psql) và pgAdmin để truy vấn cơ sở dữ liệu ChEMBL 35.
    - Viết các câu lệnh SQL để lọc, chọn và kết hợp dữ liệu cần thiết (ví dụ: cấu trúc hóa học, hoạt tính sinh học, thông tin về protein mục tiêu).
    - Lưu kết quả truy vấn vào các file CSV (Comma Separated Values) để dễ dàng xử lý bằng Python.
- **Giai đoạn 2: Tiền xử lý và Phân tích Dữ liệu bằng RDKit và scikit-learn (Data Preprocessing and Analysis using RDKit and scikit-learn)**
    - Sử dụng RDKit để xử lý cấu trúc hóa học (ví dụ: tính toán descriptor phân tử, chuyển đổi SMILES sang các định dạng khác).
    - Sử dụng scikit-learn để xây dựng các mô hình học máy (ví dụ: dự đoán hoạt tính, phân loại hợp chất).
    - Sử dụng Jupyter Notebook để viết và chạy code Python, trực quan hóa kết quả.

Mô hình phân tích chi tiết hơn:

1. **Data Acquisition:** Connect to the ChEMBL database using SQL queries to extract relevant information about compounds and their bioactivity. This includes compound structures (SMILES strings), activity values (IC50, Ki, etc.), and target information.
2. **Data Cleaning and Preprocessing:** Handle missing data, convert activity values to a consistent unit (e.g., pIC50), and filter out irrelevant or unreliable data points. Address the error related to the `numeric ~ unknown` operator by ensuring consistent data types and proper casting.
3. **Feature Engineering:** Use RDKit to generate molecular descriptors from SMILES strings. These descriptors represent various physicochemical properties of the molecules, such as molecular weight, LogP, hydrogen bond donors/acceptors, etc.
4. **Model Building and Evaluation:** Train machine learning models (e.g., regression, classification) using the generated features and activity data. Evaluate the models using appropriate metrics (e.g., RMSE, R-squared for regression; accuracy, precision, recall for classification). Address the `squared=False` error by updating scikit-learn or using `squared=True` if necessary.
5. **Interpretation and Visualization:** Analyze the model results to identify important features that contribute to bioactivity. Visualize the data and model predictions to gain insights into structure-activity relationships (SAR).

2. Hướng dẫn song ngữ (Bilingual Instructions)

English:

This guide provides step-by-step instructions for analyzing ChEMBL 35 data using RDKit and scikit-learn. It covers data extraction, preprocessing, feature engineering, model building, and evaluation. Code examples in both SQL and Python are provided.

Tiếng Việt:

Hướng dẫn này cung cấp các bước chi tiết để phân tích dữ liệu ChEMBL 35 bằng RDKit và scikit-learn. Nó bao gồm trích xuất dữ liệu, tiền xử lý, trích xuất đặc trưng, xây dựng mô hình và đánh giá. Các ví dụ mã bằng cả SQL và Python được cung cấp.

3. Code mẫu SQL và Python (SQL and Python Code Examples)

SQL (Ví dụ 1: Lấy thông tin cơ bản về hợp chất và hoạt tính)

```sql
-- Lấy 100 hợp chất có hoạt tính trên một mục tiêu cụ thể (ví dụ: CHEMBL205 - Tyrosine-protein kinase)
-- Get 100 compounds with activity against a specific target (e.g., CHEMBL205 - Tyrosine-protein kinase)
SELECT
    cmp.chembl_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    mol.molfile
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
JOIN
    molecule_dictionary mol ON cmp.molregno = mol.molregno
WHERE
    act.target_chembl_id = 'CHEMBL205'
    AND act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
LIMIT 100;
```

SQL (Ví dụ 2: Lọc các giá trị hoạt tính số)

```sql
-- Lọc các hoạt động chỉ với các giá trị số
-- Filter activities with only numeric values
SELECT
    act.molregno,
    act.standard_value
FROM
    activities act
WHERE
    act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_value ~ '^[0-9\.]+$' -- Sửa lỗi: Đảm bảo giá trị là số
LIMIT 100;
```

Khắc phục lỗi "operator does not exist: numeric ~ unknown": Lỗi này xảy ra do bạn đang cố gắng sử dụng toán tử ~ (regular expression match) trên một cột kiểu `numeric`. Để khắc phục, bạn cần đảm bảo cột `act.standard_value` được ép kiểu về `text` trước khi so sánh với regular expression. Tuy nhiên, cách tốt nhất là kiểm tra xem giá trị có phải là số bằng các hàm số học của SQL.

SQL (Ví dụ 3: Sửa lỗi "operator does not exist")

```sql
-- Lọc các hoạt động chỉ với các giá trị số (sửa lỗi)
-- Filter activities with only numeric values (error fixed)
SELECT
    act.molregno,
    act.standard_value
FROM
    activities act
WHERE
    act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$'  -- Ép kiểu về text trước khi so sánh
LIMIT 100;
```

Hoặc, bạn có thể sử dụng một cách tiếp cận khác để kiểm tra xem giá trị có phải là số mà không cần sử dụng regular expression:

```sql
-- Lọc các hoạt động chỉ với các giá trị số (cách khác)
-- Filter activities with only numeric values (alternative approach)
SELECT
    act.molregno,
    act.standard_value
FROM
    activities act
WHERE
    act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0  -- Đảm bảo giá trị lớn hơn 0 (giả định IC50 luôn
dương)
LIMIT 100;
```

Python (Ví dụ 1: Đọc dữ liệu CSV và tính toán descriptor)

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

# Đường dẫn cơ sở của dự án (thay đổi nếu cần)
# Base path of the project (change if needed)
base_path = "../data"  # Thay đổi đường dẫn nếu cần / Change the path if needed

# Đọc file CSV (ví dụ)
# Read CSV file (example)
csv_file = os.path.join(base_path, "chembl_data.csv")  # Thay đổi tên file nếu cần /
Change the filename if needed
df = pd.read_csv(csv_file)

# Hàm tính toán Molecular Weight
# Function to calculate Molecular Weight
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Áp dụng hàm tính toán MW cho cột 'smiles' (giả sử cột này chứa SMILES)
# Apply the MW calculation function to the 'smiles' column (assuming this column
contains SMILES)
```

```
df['molecular_weight'] = df['smiles'].apply(calculate_mw)

print(df.head())
```

Python (Ví dụ 2: Xây dựng mô hình dự đoán hoạt tính)

```
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
import os

# Đường dẫn cơ sở của dự án (thay đổi nếu cần)
# Base path of the project (change if needed)
base_path = "../data"  # Thay đổi đường dẫn nếu cần / Change the path if needed

# Đọc file CSV (ví dụ)
# Read CSV file (example)
csv_file = os.path.join(base_path, "chembl_data.csv")  # Thay đổi tên file nếu cần /
Change the filename if needed
df = pd.read_csv(csv_file)

# Tiền xử lý dữ liệu (ví dụ: chuyển đổi IC50 sang pIC50)
# Data preprocessing (e.g., convert IC50 to pIC50)
df = df.dropna(subset=['standard_value']) # Loại bỏ các hàng có giá trị
'standard_value' bị thiếu / Remove rows with missing 'standard_value'

# Define pIC50 transformation function
def pIC50(IC50_nM):
    pIC50 = -np.log10(IC50_nM*(10**-9))
    return pIC50

df['pIC50'] = df['standard_value'].apply(pIC50) # Apply pIC50 transformation function
df = df.replace([np.inf, -np.inf], np.nan) # Replace infinity values with NaN
df = df.dropna(subset=['pIC50']) # Drop rows with NaN values

# Hàm tính toán descriptor (ví dụ: Molecular Weight)
# Function to calculate descriptors (e.g., Molecular Weight)
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

df['molecular_weight'] = df['smiles'].apply(calculate_mw) # Calculate molecular weight
df = df.dropna(subset=['molecular_weight']) # Drop rows with missing MW

# Chọn features và target
# Select features and target
X = df[['molecular_weight']]  # Ví dụ: chỉ sử dụng MW làm feature / Example: use only
MW as a feature
y = df['pIC50']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
# 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)

# Xây dựng mô hình Linear Regression
# Build Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
# Predict on the test set
y_pred = model.predict(X_test)

# Đánh giá mô hình
# 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}")
```

Khắc phục lỗi "squared=False": Nếu bạn gặp lỗi này, hãy đảm bảo rằng bạn đang sử dụng phiên bản scikit-learn mới nhất. Nếu không thể nâng cấp, bạn có thể bỏ tham số `squared=False` (nó chỉ ảnh hưởng đến việc trả về MSE hay RMSE).

4. Thêm 3 ví dụ code SQL và Python mẫu (3 More SQL and Python Code Examples)

SQL (Ví dụ 4: Lấy số lượng hợp chất cho mỗi target)

```sql
-- Lấy số lượng hợp chất có trong mỗi mục tiêu
-- Get the number of compounds for each target
SELECT
    act.target_chembl_id,
    COUNT(DISTINCT cmp.molregno) AS num_compounds
FROM
    activities act
JOIN
    compound_structures cmp ON act.molregno = cmp.molregno
GROUP BY
    act.target_chembl_id
ORDER BY
    num_compounds DESC
LIMIT 100;
```

SQL (Ví dụ 5: Tìm các hợp chất có hoạt tính trên nhiều mục tiêu)

```sql
-- Tìm các hợp chất có hoạt tính trên nhiều mục tiêu
-- Find compounds that are active against multiple targets
SELECT
    cmp.chembl_id,
    COUNT(DISTINCT act.target_chembl_id) AS num_targets
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
GROUP BY
    cmp.chembl_id
HAVING
    COUNT(DISTINCT act.target_chembl_id) > 1
ORDER BY
    num_targets DESC
LIMIT 100;
```

Python (Ví dụ 3: Tính toán LogP bằng RDKit)

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

# Đường dẫn cơ sở của dự án (thay đổi nếu cần)
# Base path of the project (change if needed)
base_path = "../data"  # Thay đổi đường dẫn nếu cần / Change the path if needed

# Đọc file CSV (ví dụ)
# Read CSV file (example)
csv_file = os.path.join(base_path, "chembl_data.csv")  # Thay đổi tên file nếu cần /
Change the filename if needed
df = pd.read_csv(csv_file)

# Hàm tính toán LogP
# Function to calculate LogP
def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Crippen.MolLogP(mol)
    else:
        return None

# Áp dụng hàm tính toán LogP cho cột 'smiles'
# Apply the LogP calculation function to the 'smiles' column
df['logp'] = df['smiles'].apply(calculate_logp)

print(df.head())
```

Python (Ví dụ 4: Trực quan hóa kết quả với matplotlib)

```python
import pandas as pd
import matplotlib.pyplot as plt
import os

# Đường dẫn cơ sở của dự án (thay đổi nếu cần)
# Base path of the project (change if needed)
base_path = "../data"  # Thay đổi đường dẫn nếu cần / Change the path if needed

# Đọc file CSV (ví dụ)
# Read CSV file (example)
csv_file = os.path.join(base_path, "chembl_data.csv")  # Thay đổi tên file nếu cần /
Change the filename if needed
df = pd.read_csv(csv_file)

# Vẽ biểu đồ scatter giữa Molecular Weight và pIC50 (ví dụ)
# Create a scatter plot between Molecular Weight and pIC50 (example)
plt.scatter(df['molecular_weight'], df['pIC50'])
plt.xlabel('Molecular Weight')
plt.ylabel('pIC50')
plt.title('Molecular Weight vs pIC50')
plt.show()
```

Python (Ví dụ 5: Tính toán TPSA - Topological Polar Surface Area)

```python
import pandas as pd
from rdkit import Chem
```

```
from rdkit.Chem import Descriptors3D
import os

# Đường dẫn cơ sở của dự án (thay đổi nếu cần)
# Base path of the project (change if needed)
base_path = "../data"  # Thay đổi đường dẫn nếu cần / Change the path if needed

# Đọc file CSV (ví dụ)
# Read CSV file (example)
csv_file = os.path.join(base_path, "chembl_data.csv")  # Thay đổi tên file nếu cần /
Change the filename if needed
df = pd.read_csv(csv_file)

# Hàm tính toán TPSA
# Function to calculate TPSA
def calculate_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors3D.TPSA(mol)
    else:
        return None

# Áp dụng hàm tính toán TPSA cho cột 'smiles'
# Apply the TPSA calculation function to the 'smiles' column
df['tpsa'] = df['smiles'].apply(calculate_tpsa)

print(df.head())
```

Lưu ý quan trọng:

- Hãy nhớ thay đổi đường dẫn file và tên file CSV cho phù hợp với cấu trúc thư mục và dữ liệu của bạn.
- Các ví dụ trên chỉ là điểm khởi đầu. Bạn cần điều chỉnh code cho phù hợp với mục tiêu nghiên cứu cụ thể của mình.
- Hãy luôn kiểm tra và làm sạch dữ liệu trước khi phân tích để đảm bảo kết quả chính xác.
- Nếu bạn gặp bất kỳ vấn đề nào khác, đừng ngần ngại hỏi!

Chúc bạn thành công với dự án của mình!