PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Bản đồ không gian hóa học của các phân tử ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu Chembl 35 này. Với vai trò là Chuyên gia R&D Dược học, tôi sẽ cung cấp hướng dẫn chi tiết, code mẫu và giải thích rõ ràng để bạn có thể triển khai dự án một cách hiệu quả.

1. Phân Tích Mô Hình Phân Tích (Analysis of the Analysis Model)

Dựa trên yêu cầu của bạn, chúng ta sẽ xây dựng một quy trình phân tích dữ liệu Chembl 35 kết hợp RDKit để hỗ trợ nghiên cứu và phát triển thuốc. Mô hình này bao gồm các bước chính sau:

- **Thu thập và Chuẩn bị Dữ liệu (Data Acquisition and Preparation):**
  - Sử dụng SQL để truy vấn dữ liệu từ database Chembl 35 (trên PostgreSQL).
  - Lọc và tiền xử lý dữ liệu (ví dụ: loại bỏ các giá trị không hợp lệ, chuẩn hóa dữ liệu).
  - Lưu dữ liệu đã xử lý vào file CSV.
- **Tính Toán Đặc Trưng Phân Tử (Molecular Feature Calculation):**
  - Sử dụng RDKit để tính toán các đặc trưng phân tử (molecular descriptors) từ SMILES strings. Các đặc trưng này có thể bao gồm:
    - Tính chất vật lý hóa học (physicochemical properties): LogP, Molecular Weight, Topological Polar Surface Area (TPSA).
    - Đặc trưng cấu trúc (structural features): số lượng vòng, số lượng nguyên tử, số lượng liên kết.
    - Fingerprints: Morgan fingerprints, MACCS keys.
- **Phân Tích Thống Kê và Trực Quan Hóa (Statistical Analysis and Visualization):**
  - Sử dụng Python (với các thư viện như pandas, numpy, matplotlib, seaborn) để thực hiện phân tích thống kê mô tả (descriptive statistics) và trực quan hóa dữ liệu.
  - Ví dụ: vẽ biểu đồ phân phối của các đặc trưng phân tử, biểu đồ scatter để tìm mối tương quan giữa các đặc trưng và hoạt tính sinh học.
- **Xây Dựng Mô Hình Học Máy (Machine Learning Model Building):**
  - Sử dụng scikit-learn để xây dựng các mô hình học máy dự đoán hoạt tính sinh học (ví dụ: hồi quy tuyến tính, random forest, support vector machines).
  - Sử dụng các đặc trưng phân tử làm đầu vào cho mô hình.
  - Đánh giá hiệu suất mô hình bằng các metrics phù hợp (ví dụ: R-squared, RMSE, AUC).

2. Hướng Dẫn Song Ngữ (Bilingual Instructions)

2.1. Kết Nối Cơ Sở Dữ Liệu và Truy Vấn SQL (Database Connection and SQL Queries)

Tiếng Việt:

1. Sử dụng pgAdmin để kết nối đến cơ sở dữ liệu PostgreSQL của bạn.
2. Tạo một query mới và viết các câu lệnh SQL để truy vấn dữ liệu mong muốn từ Chembl 35.
3. Lưu kết quả truy vấn vào file CSV (ví dụ: `../data/chembl_data.csv`).

English:

1. Use pgAdmin to connect to your PostgreSQL database.
2. Create a new query and write SQL statements to retrieve the desired data from Chembl 35.
3. Save the query results to a CSV file (e.g., `../data/chembl_data.csv`).

2.2. Tính Toán Đặc Trưng Phân Tử với RDKit (Molecular Feature Calculation with RDKit)

Tiếng Việt:

1. Trong Jupyter Notebook, import thư viện RDKit.
2. Đọc file CSV chứa dữ liệu Chembl.
3. Sử dụng RDKit để chuyển đổi SMILES strings thành các đối tượng phân tử.
4. Tính toán các đặc trưng phân tử mong muốn (ví dụ: LogP, Molecular Weight).
5. Lưu các đặc trưng này vào DataFrame.

English:

1. In a Jupyter Notebook, import the RDKit library.
2. Read the CSV file containing Chembl data.
3. Use RDKit to convert SMILES strings into molecule objects.
4. Calculate the desired molecular features (e.g., LogP, Molecular Weight).
5. Store these features in a DataFrame.

2.3. Xây Dựng Mô Hình Học Máy (Machine Learning Model Building)

Tiếng Việt:

1. Chuẩn bị dữ liệu: tách features (đặc trưng) và target (biến mục tiêu).
2. Chia dữ liệu thành tập huấn luyện và tập kiểm tra.
3. Chọn một mô hình học máy (ví dụ: Random Forest).
4. Huấn luyện mô hình trên tập huấn luyện.
5. Đánh giá mô hình trên tập kiểm tra.

English:

1. Prepare the data: separate features and target.
2. Split the data into training and test sets.
3. Choose a machine learning model (e.g., Random Forest).
4. Train the model on the training set.
5. Evaluate the model on the test set.

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

3.1. Code SQL

```sql
-- English: Select 100 rows from the activities table with specific conditions
-- Vietnamese: Chọn 100 dòng từ bảng activities với các điều kiện cụ thể
SELECT
    act.molregno,
    act.standard_type,
    act.standard_relation,
    act.standard_value,
    act.standard_units,
    cmp.chembl_id,
    cmp.smiles
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
```

```
    --AND act.standard_value ~ '^[0-9\.]+$' -- This line might cause an error
LIMIT 100;
```

Lưu ý: Dòng `AND act.standard_value ~ '^[0-9\.]+$'` có thể gây ra lỗi vì kiểu dữ liệu của `standard_value` là numeric. Bạn có thể bỏ qua dòng này hoặc chuyển đổi `standard_value` sang kiểu text để so sánh.

3.2. Code Python

```
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Define the base path for your project
base_path = "."  # Assuming the notebook is in the root directory

# Read the CSV file containing Chembl data
data_path = os.path.join(base_path, 'data', 'chembl_data.csv') # Assuming the CSV is
stored in a 'data' folder
df = pd.read_csv(data_path)

# Function to calculate molecular weight using RDKit
def calculate_molecular_weight(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol)
        else:
            return None
    except:
        return None

# Apply the function to calculate molecular weight for each SMILES string
df['molecular_weight'] = df['smiles'].apply(calculate_molecular_weight)

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

# Drop rows with missing values (NaN)
df = df.dropna()

# Define features (X) and target (y)
X = df[['molecular_weight']]
y = df['standard_value']

# Split the 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)

# Create a Random Forest Regressor model
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
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}")
```

4. Ví Dụ Code SQL và Python (SQL and Python Code Examples)

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

Ví dụ 1: Truy vấn thông tin cơ bản về các hợp chất (Basic Compound Information Query)

SQL:

```
-- English: Select chembl_id and smiles for the first 100 compounds
-- Vietnamese: Chọn chembl_id và smiles của 100 hợp chất đầu tiên
SELECT chembl_id, smiles FROM molecule_dictionary LIMIT 100;
```

Python:

```
# English: Read the first 100 rows from the molecule_dictionary table into a Pandas DataFrame
# Vietnamese: Đọc 100 dòng đầu tiên từ bảng molecule_dictionary vào DataFrame Pandas
import pandas as pd
import psycopg2

# Database credentials
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35'
}

# Connect to the database
conn = psycopg2.connect(**db_params)

# SQL query to retrieve the first 100 rows from molecule_dictionary
sql_query = "SELECT chembl_id, smiles FROM molecule_dictionary LIMIT 100;"

# Read the data into a Pandas DataFrame
df = pd.read_sql(sql_query, conn)

# Close the database connection
conn.close()

# Print the DataFrame
print(df.head())
```

Ví dụ 2: Tính LogP sử dụng RDKit (Calculate LogP using RDKit)

Python:

```
# English: Calculate LogP for a list of SMILES strings
# Vietnamese: Tính LogP cho một danh sách các chuỗi SMILES
from rdkit import Chem
from rdkit.Chem import Crippen

smiles_list = ['CC(=O)Oc1ccccc1C(=O)O', 'c1ccccc1', 'CCO']
```

```
logp_values = []
for smiles in smiles_list:
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        logp = Crippen.MolLogP(mol)
        logp_values.append(logp)
    else:
        logp_values.append(None)

print(logp_values)
```

Ví dụ 3: Lọc các hợp chất có trọng lượng phân tử nhất định (Filter Compounds by Molecular Weight)

SQL:

```
-- English: This SQL query is not directly possible as molecular weight is not stored in the database.
-- You would need to calculate it in Python using RDKit after fetching SMILES.
-- Vietnamese: Truy vấn SQL này không thể thực hiện trực tiếp vì trọng lượng phân tử không được lưu trữ trong cơ sở dữ liệu.
-- Bạn cần tính toán nó trong Python bằng RDKit sau khi lấy SMILES.

-- Example: Select chembl_id and smiles for compounds, then calculate molecular weight in Python
-- Ví dụ: Chọn chembl_id và smiles cho các hợp chất, sau đó tính trọng lượng phân tử trong Python
SELECT chembl_id, smiles FROM molecule_dictionary LIMIT 100;
```

Python:

```
# English: Filter compounds by molecular weight after fetching data
# Vietnamese: Lọc các hợp chất theo trọng lượng phân tử sau khi lấy dữ liệu
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import psycopg2

# Database credentials
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35'
}

# Connect to the database
conn = psycopg2.connect(**db_params)

# SQL query to retrieve chembl_id and smiles
sql_query = "SELECT chembl_id, smiles FROM molecule_dictionary LIMIT 100;"

# Read the data into a Pandas DataFrame
df = pd.read_sql(sql_query, conn)

# Close the database connection
conn.close()

# Function to calculate molecular weight
def calculate_molecular_weight(smiles):
```

```
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol)
        else:
            return None
    except:
        return None

# Apply the function to calculate molecular weight for each SMILES string
df['molecular_weight'] = df['smiles'].apply(calculate_molecular_weight)

# Filter compounds with molecular weight between 200 and 400
filtered_df = df[(df['molecular_weight'] >= 200) & (df['molecular_weight'] <= 400)]

print(filtered_df)
```

Ví dụ 4: Phân tích hoạt tính sinh học (Activity Analysis)

SQL:

```
-- English: Select IC50 values for a specific target
-- Vietnamese: Chọn giá trị IC50 cho một target cụ thể
SELECT act.standard_value
FROM activities act
JOIN target_dictionary tgt ON act.tid = tgt.tid
WHERE tgt.chembl_id = 'CHEMBL205'  -- Replace with the desired target CHEMBL ID
AND act.standard_type = 'IC50'
AND act.standard_relation = '='
LIMIT 100;
```

Python:

```
# English: Analyze IC50 values fetched from the database
# Vietnamese: Phân tích các giá trị IC50 được lấy từ cơ sở dữ liệu
import pandas as pd
import matplotlib.pyplot as plt
import psycopg2

# Database credentials
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35'
}

# Connect to the database
conn = psycopg2.connect(**db_params)

# SQL query to retrieve IC50 values for a specific target
sql_query = """
SELECT act.standard_value
FROM activities act
JOIN target_dictionary tgt ON act.tid = tgt.tid
WHERE tgt.chembl_id = 'CHEMBL205'  -- Replace with the desired target CHEMBL ID
AND act.standard_type = 'IC50'
AND act.standard_relation = '='
LIMIT 100;
"""
```

```python
# Read the data into a Pandas DataFrame
df = pd.read_sql(sql_query, conn)

# Close the database connection
conn.close()

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

# Drop rows with missing values (NaN)
df = df.dropna()

# Plot a histogram of IC50 values
plt.hist(df['standard_value'], bins=20)
plt.xlabel('IC50 Value')
plt.ylabel('Frequency')
plt.title('Distribution of IC50 Values')
plt.show()
```

Ví dụ 5: Xây dựng mô hình dự đoán hoạt tính (Activity Prediction Model)

Python:

```python
# English: Build a simple activity prediction model using molecular weight
# Vietnamese: Xây dựng mô hình dự đoán hoạt tính đơn giản sử dụng trọng lượng phân tử
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 psycopg2

# Database credentials
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35'
}

# Connect to the database
conn = psycopg2.connect(**db_params)

# SQL query to retrieve data for activity prediction
sql_query = """
SELECT act.standard_value, cmp.smiles
FROM activities act
JOIN molecule_dictionary cmp ON act.molregno = cmp.molregno
WHERE act.standard_type = 'IC50'
AND act.standard_relation = '='
AND act.standard_value IS NOT NULL
LIMIT 100;
"""

# Read the data into a Pandas DataFrame
df = pd.read_sql(sql_query, conn)

# Close the database connection
conn.close()
```

```python
# Function to calculate molecular weight
def calculate_molecular_weight(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol)
        else:
            return None
    except:
        return None

# Apply the function to calculate molecular weight for each SMILES string
df['molecular_weight'] = df['smiles'].apply(calculate_molecular_weight)

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

# Drop rows with missing values (NaN)
df = df.dropna()

# Prepare data for modeling
X = df[['molecular_weight']]
y = df['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)

# Create a linear regression model
model = LinearRegression()

# Train the model
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)
print(f"Mean Squared Error: {mse}")
```

Lưu ý quan trọng:

- **Sửa lỗi SQL:** Trong truy vấn SQL, hãy đảm bảo xử lý đúng kiểu dữ liệu của cột `standard_value`. Nếu cần, bạn có thể bỏ qua việc lọc theo regular expression hoặc chuyển đổi kiểu dữ liệu trước khi so sánh.
- **Phiên bản Scikit-learn:** Nếu bạn gặp lỗi về tham số `squared=False` trong `mean_squared_error`, hãy cập nhật scikit-learn lên phiên bản mới nhất hoặc bỏ tham số này (phiên bản cũ mặc định `squared=True`).
- **Đường dẫn:** Sử dụng `os.path.join(base_path, ...)` để đảm bảo tính di động của code khi di chuyển giữa các môi trường.

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