PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Phân nhóm target theo tương đồng chuỗi

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit. Với vai trò là một chuyên gia R&D dược học, tôi hiểu tầm quan trọng của việc khai thác dữ liệu lớn để thúc đẩy quá trình phát triển thuốc. Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu và các ví dụ để bạn bắt đầu dự án **Topic_CheMBL_35_33** một cách hiệu quả.

1. Phân tích Mô hình (Analysis Model)

- **Mục tiêu:** Phân tích mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học của các hợp chất trong cơ sở dữ liệu ChEMBL 35.
- **Phương pháp:**
    - **Trích xuất dữ liệu:** Sử dụng SQL để truy vấn và trích xuất thông tin từ cơ sở dữ liệu ChEMBL 35, bao gồm cấu trúc hóa học (SMILES), hoạt tính sinh học (IC50, Ki, Kd, EC50), và các thông tin liên quan khác.
    - **Biểu diễn cấu trúc:** Sử dụng RDKit để chuyển đổi chuỗi SMILES thành các biểu diễn số (fingerprints, descriptors) có thể sử dụng trong các mô hình học máy.
    - **Phân tích tương quan:** Tìm kiếm mối tương quan giữa các thuộc tính cấu trúc và hoạt tính sinh học. Điều này có thể bao gồm việc sử dụng các phương pháp thống kê như hồi quy tuyến tính, hoặc các mô hình học máy phức tạp hơn như random forests, support vector machines (SVM), hoặc neural networks.
    - **Xây dựng mô hình dự đoán:** Phát triển các mô hình có khả năng dự đoán hoạt tính sinh học của các hợp chất mới dựa trên cấu trúc của chúng.
    - **Đánh giá mô hình:** Sử dụng các kỹ thuật đánh giá mô hình phù hợp (ví dụ: cross-validation, ROC AUC, RMSE) để đánh giá hiệu suất của các mô hình đã xây dựng.
- **Analysis Model:** Analyzing the relationship between chemical structure and biological activity of compounds in the ChEMBL 35 database.
- **Method:**
    - **Data Extraction:** Use SQL to query and extract information from the ChEMBL 35 database, including chemical structures (SMILES), biological activities (IC50, Ki, Kd, EC50), and other relevant information.
    - **Structure Representation:** Use RDKit to convert SMILES strings into numerical representations (fingerprints, descriptors) that can be used in machine learning models.
    - **Correlation Analysis:** Search for correlations between structural properties and biological activities. This can include using statistical methods such as linear regression, or more complex machine learning models such as random forests, support vector machines (SVM), or neural networks.
    - **Predictive Model Building:** Develop models capable of predicting the biological activity of new compounds based on their structure.
    - **Model Evaluation:** Use appropriate model evaluation techniques (e.g., cross-validation, ROC AUC, RMSE) to assess the performance of the constructed models.

2. Hướng dẫn song ngữ (Bilingual Instructions)

a. Kết nối đến cơ sở dữ liệu ChEMBL 35 (Connecting to ChEMBL 35 Database)

- **Tiếng Việt:** Sử dụng thư viện psycopg2 trong Python để kết nối đến cơ sở dữ liệu PostgreSQL. Cung cấp thông tin đăng nhập (IP, user, password, database) mà bạn đã cung cấp.
- **English:** Use the psycopg2 library in Python to connect to the PostgreSQL database. Provide the login information (IP, user, password, database) you provided.

```python
import psycopg2

# Database credentials
db_host = "192.168.206.136"
db_user = "rd"
db_pass = "rd"
db_name = "chembl_35"

try:
    # Establish connection
    conn = psycopg2.connect(host=db_host, user=db_user, password=db_pass,
database=db_name)
    cursor = conn.cursor()
    print("Connected to the database successfully!")

except psycopg2.Error as e:
    print(f"Error connecting to the database: {e}")
```

b. Trích xuất dữ liệu sử dụng SQL (Extracting data using SQL)

- **Tiếng Việt:** Viết các câu truy vấn SQL để lấy dữ liệu từ các bảng trong cơ sở dữ liệu ChEMBL 35. Lưu ý giới hạn số lượng dòng dữ liệu để tránh quá tải.
- **English:** Write SQL queries to retrieve data from tables in the ChEMBL 35 database. Remember to limit the number of rows to avoid overloading.

```sql
-- Lấy thông tin từ bảng compounds và activities (ví dụ)
-- Get information from the compounds and activities tables (example)
SELECT
    cmp.chembl_id,
    cmp.smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
LIMIT 100;
```

c. Tiền xử lý dữ liệu với RDKit (Data preprocessing with RDKit)

- **Tiếng Việt:** Sử dụng RDKit để chuyển đổi SMILES thành các đối tượng phân tử, tính toán fingerprints hoặc descriptors, và xử lý các vấn đề về dữ liệu (ví dụ: loại bỏ các phân tử không hợp lệ).
- **English:** Use RDKit to convert SMILES into molecular objects, calculate fingerprints or descriptors, and handle data issues (e.g., removing invalid molecules).

```python
from rdkit import Chem
from rdkit.Chem import AllChem
import numpy as np

def calculate_fingerprint(smiles):
    """Calculates Morgan fingerprint for a given SMILES string."""
    mol = Chem.MolFromSmiles(smiles)
    if mol:
```

```
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
        return np.array(fp)
    else:
        return None
```

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

a. SQL (ví dụ lấy dữ liệu IC50, giới hạn 100 dòng):

```
SELECT
    cmp.chembl_id,
    cmp.smiles,
    act.standard_value
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL  -- Loại bỏ giá trị NULL
  AND act.standard_value > 0       -- Loại bỏ giá trị âm hoặc bằng 0
  AND act.standard_value::text ~ '^[0-9\.]+$' -- Chỉ lấy giá trị số
LIMIT 100;
```

Lưu ý về lỗi SQL:

- Lỗi `ERROR: operator does not exist: numeric ~ unknown` xảy ra do bạn đang cố gắng sử dụng toán tử `~` (regular expression match) trên một cột kiểu số (`numeric`). Để khắc phục, bạn cần ép kiểu cột `act.standard_value` về kiểu `text` trước khi so sánh với regular expression.

b. Python (ví dụ tính toán fingerprint và chuẩn bị dữ liệu cho mô hình):

```
import psycopg2
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import os

# Database credentials
db_host = "192.168.206.136"
db_user = "rd"
db_pass = "rd"
db_name = "chembl_35"

# SQL query to fetch data
sql_query = """
SELECT
    cmp.chembl_id,
    cmp.smiles,
    act.standard_value
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
```

```
  AND act.standard_value > 0
  AND act.standard_value::text ~ '^[0-9\.]+$'
LIMIT 100;
"""

# Function to calculate Morgan fingerprint
def calculate_fingerprint(smiles):
    """Calculates Morgan fingerprint for a given SMILES string."""
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
        return np.array(fp)
    else:
        return None

# Main execution block
if __name__ == "__main__":
    try:
        # Establish connection
        conn = psycopg2.connect(host=db_host, user=db_user, password=db_pass,
database=db_name)
        cursor = conn.cursor()

        # Fetch data into a Pandas DataFrame
        df = pd.read_sql_query(sql_query, conn)

        # Close the connection
        conn.close()

        # Calculate fingerprints
        df['fingerprint'] = df['smiles'].apply(calculate_fingerprint)

        # Drop rows with missing fingerprints
        df.dropna(subset=['fingerprint'], inplace=True)

        # Convert standard_value to numeric
        df['standard_value'] = pd.to_numeric(df['standard_value'])

        # Prepare data for machine learning
        X = np.stack(df['fingerprint'].values)
        y = np.log10(df['standard_value'].values) # Use log10 for better distribution

        # 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)

        # Train a Random Forest Regressor model
        model = RandomForestRegressor(n_estimators=100, random_state=42)
        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}")
```

```
    except psycopg2.Error as e:
        print(f"Error: {e}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

Lưu ý về lỗi Python:

- **Scikit-learn version:** Nếu bạn gặp lỗi về tham số `squared=False` trong `mean_squared_error`, hãy cập nhật scikit-learn lên phiên bản mới nhất bằng lệnh `pip install -U scikit-learn`. Nếu không thể cập nhật, hãy loại bỏ tham số `squared=False` (phiên bản cũ mặc định trả về MSE, không phải RMSE).
- **Data Types:** Đảm bảo kiểu dữ liệu của các cột là chính xác. Ví dụ: `standard_value` phải là kiểu số để thực hiện các phép tính toán.
- **Missing Data:** Xử lý các giá trị `NULL` hoặc `NaN` trong dữ liệu. Bạn có thể loại bỏ các dòng chứa giá trị thiếu hoặc sử dụng các phương pháp imputation để điền giá trị.
- **Log Transformation:** Áp dụng log transformation cho giá trị `standard_value` (ví dụ: IC50) để làm giảm độ lệch của dữ liệu và cải thiện hiệu suất mô hình.

4. 5 Ví dụ Code SQL và Python (5 SQL and Python Code Examples)

Dưới đây là 5 ví dụ code SQL và Python để minh họa các tác vụ khác nhau trong quá trình phân tích dữ liệu ChEMBL 35.

Ví dụ 1: Lấy dữ liệu Ki cho một mục tiêu cụ thể (Retrieve Ki data for a specific target)

SQL:

```sql
SELECT
    cmp.chembl_id,
    cmp.smiles,
    act.standard_value
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE act.standard_type = 'Ki'
  AND act.standard_units = 'nM'
  AND td.chembl_id = 'CHEMBL205'  -- Ví dụ: EGFR
LIMIT 100;
```

Python:

```
import pandas as pd
import psycopg2

def get_ki_data(target_chembl_id, limit=100):
    conn = psycopg2.connect(host=db_host, user=db_user, password=db_pass,
database=db_name)
    sql_query = f"""
    SELECT
        cmp.chembl_id,
        cmp.smiles,
        act.standard_value
    FROM
        compound_structures cmp
    JOIN
        activities act ON cmp.molregno = act.molregno
    JOIN
```

```
        target_dictionary td ON act.tid = td.tid
    WHERE act.standard_type = 'Ki'
      AND act.standard_units = 'nM'
      AND td.chembl_id = '{target_chembl_id}'
    LIMIT {limit};
    """
    df = pd.read_sql_query(sql_query, conn)
    conn.close()
    return df

# Example usage
target = 'CHEMBL205'
ki_data = get_ki_data(target)
print(ki_data.head())
```

Ví dụ 2: Tính toán các descriptor vật lý hóa học (Calculate physicochemical descriptors)

Python:

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

def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        mw = Descriptors.MolWt(mol)
        logp = Descriptors.MolLogP(mol)
        hbd = Descriptors.NumHDonors(mol)
        hba = Descriptors.NumHAcceptors(mol)
        return mw, logp, hbd, hba
    else:
        return None, None, None, None

# Example usage (assuming you have a DataFrame 'df' with a 'smiles' column)
df[['mw', 'logp', 'hbd', 'hba']] = df['smiles'].apply(lambda x:
pd.Series(calculate_descriptors(x)))
print(df[['smiles', 'mw', 'logp', 'hbd', 'hba']].head())
```

Ví dụ 3: Xây dựng mô hình hồi quy tuyến tính (Build a linear regression model)

Python:

```
from sklearn.linear_model import LinearRegression

# Assuming you have X_train, X_test, y_train, y_test from the previous example

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

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Linear Regression - Mean Squared Error: {mse}")
print(f"Linear Regression - R-squared: {r2}")
```

Ví dụ 4: Tìm kiếm các hợp chất tương tự (Similarity Search)

Python:

```python
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.DataStructs import FingerprintSimilarity
import numpy as np

def find_similar_compounds(smiles, df, threshold=0.7):
    """Finds compounds in a DataFrame similar to a given SMILES string."""
    query_mol = Chem.MolFromSmiles(smiles)
    if not query_mol:
        return []

    query_fp = AllChem.GetMorganFingerprintAsBitVect(query_mol, 2, nBits=2048)
    similar_compounds = []

    for index, row in df.iterrows():
        mol = Chem.MolFromSmiles(row['smiles'])
        if mol:
            fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
            similarity = FingerprintSimilarity(query_fp, fp)
            if similarity >= threshold:
                similar_compounds.append((row['chembl_id'], similarity))

    return similar_compounds

# Example usage
reference_smiles = "Cc1ccccc1" # Example: Toluene
similar_compounds = find_similar_compounds(reference_smiles, df)
print(f"Compounds similar to {reference_smiles}: {similar_compounds}")
```

Ví dụ 5: Lưu trữ kết quả vào cơ sở dữ liệu (Store results in the database)

Python:

```python
import psycopg2

def store_results(chembl_id, predicted_value):
    """Stores predicted values in a table in the database."""
    conn = psycopg2.connect(host=db_host, user=db_user, password=db_pass,
database=db_name)
    cursor = conn.cursor()
    try:
        sql = "INSERT INTO predictions (chembl_id, predicted_value) VALUES (%s, %s)"
        cursor.execute(sql, (chembl_id, predicted_value))
        conn.commit()
        print(f"Successfully stored prediction for {chembl_id}")
    except psycopg2.Error as e:
        print(f"Error storing prediction: {e}")
    finally:
        conn.close()

# Example usage (assuming you have chembl_id and predicted_value)
# You would need to create a 'predictions' table in your database first.
store_results('CHEMBL123', 7.5)
```

Quan trọng:

- **AIMLOps Template:** Hãy nhớ tuân thủ cấu trúc thư mục AIMLOps khi lưu trữ code và dữ liệu.
- **Documentation:** Ghi chú code của bạn đầy đủ để người khác (và bạn trong tương lai) có thể hiểu được.

- **Experimentation:** Thử nghiệm với các phương pháp và mô hình khác nhau để tìm ra phương pháp tốt nhất cho bài toán của bạn.
- **Error Handling:** Xử lý các lỗi có thể xảy ra trong quá trình thực thi code.
- **Data Exploration:** Dành thời gian khám phá dữ liệu của bạn để hiểu rõ hơn về nó.

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