PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Giảm chiều dữ liệu cho đặc trưng hóa học

Tuyệt vời! Tôi rất sẵn lòng giúp bạn với dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Với vai trò là một chuyên gia R&D dược học, tôi hiểu tầm quan trọng của việc khai thác dữ liệu từ các cơ sở dữ liệu lớn như ChEMBL để hỗ trợ quá trình phát triển thuốc.

Dưới đây là phân tích chi tiết và hướng dẫn song ngữ, cùng với các ví dụ code SQL và Python để bạn bắt đầu:

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

Mục tiêu: Xây dựng một mô hình dự đoán hoạt tính sinh học của các hợp chất dựa trên cấu trúc hóa học của chúng, sử dụng dữ liệu từ ChEMBL 35.

Các bước chính:

1. **Thu thập và chuẩn bị dữ liệu (Data Acquisition and Preparation):**
    - Trích xuất dữ liệu cần thiết từ cơ sở dữ liệu ChEMBL 35 (sử dụng SQL).
    - Làm sạch và tiền xử lý dữ liệu:
        - Xử lý các giá trị thiếu (missing values).
        - Chuẩn hóa dữ liệu (ví dụ: chuyển đổi IC50 sang pIC50).
        - Lọc các hoạt tính không hợp lệ hoặc không đáng tin cậy.
2. **Tính toán đặc trưng hóa học (Chemical Feature Calculation):**
    - Sử dụng RDKit để tính toán các descriptor (đặc trưng) hóa học từ cấu trúc SMILES của các hợp chất. Các descriptor này có thể bao gồm:
        - Molecular weight (khối lượng phân tử)
        - LogP (hệ số phân bố octanol-nước)
        - Hydrogen bond donors/acceptors (số lượng liên kết hydro cho/nhận)
        - Topological polar surface area (TPSA)
        - Số lượng vòng (number of rings)
        - ... và nhiều descriptor khác.
3. **Lựa chọn đặc trưng (Feature Selection):**
    - Chọn các descriptor quan trọng nhất để đưa vào mô hình. Điều này có thể được thực hiện bằng các phương pháp:
        - Univariate feature selection (ví dụ: SelectKBest)
        - Recursive feature elimination (RFE)
        - Sử dụng các mô hình machine learning để đánh giá tầm quan trọng của đặc trưng (feature importance).
4. **Xây dựng mô hình (Model Building):**
    - Chọn một thuật toán machine learning phù hợp:
        - **Regression:** Nếu bạn muốn dự đoán giá trị hoạt tính liên tục (ví dụ: pIC50). Các thuật toán phổ biến bao gồm:
            - Linear Regression
            - Random Forest Regression
            - Support Vector Regression (SVR)
            - Gradient Boosting Regression (ví dụ: XGBoost, LightGBM)

        - **Classification:** Nếu bạn muốn phân loại các hợp chất thành hoạt tính/không hoạt tính. Các thuật toán phổ biến bao gồm:
            - Logistic Regression
            - Random Forest Classification
            - Support Vector Machines (SVM)
            - Gradient Boosting Classification
    - Huấn luyện mô hình trên dữ liệu huấn luyện (training data).
5. **Đánh giá mô hình (Model Evaluation):**
    - Sử dụng dữ liệu kiểm tra (test data) để đánh giá hiệu suất của mô hình.
    - Sử dụng các metric phù hợp:
        - **Regression:** R-squared, Mean Squared Error (MSE), Root Mean Squared Error (RMSE)
        - **Classification:** Accuracy, Precision, Recall, F1-score, AUC-ROC
6. **Tối ưu hóa mô hình (Model Optimization):**
    - Điều chỉnh các siêu tham số (hyperparameters) của mô hình để cải thiện hiệu suất.
    - Sử dụng các kỹ thuật như cross-validation để đánh giá và lựa chọn mô hình tốt nhất.

2. Hướng dẫn Song ngữ (Bilingual Guidance)

Data Extraction and Preparation (Trích xuất và chuẩn bị dữ liệu):

- **SQL:** Use SQL queries to extract relevant data from the ChEMBL database. This includes compound structures (SMILES), activity values (IC50, Ki, etc.), and target information. *(Sử dụng truy vấn SQL để trích xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL. Điều này bao gồm cấu trúc hợp chất (SMILES), giá trị hoạt tính (IC50, Ki, v.v.) và thông tin mục tiêu.)*
- **Python:** Load the extracted data into a Pandas DataFrame for further processing. *(Python: Tải dữ liệu đã trích xuất vào Pandas DataFrame để xử lý thêm.)*

Feature Engineering (Xây dựng đặc trưng):

- **RDKit:** Use RDKit to calculate molecular descriptors from the SMILES strings. These descriptors represent various physicochemical properties of the compounds. *(RDKit: Sử dụng RDKit để tính toán các descriptor phân tử từ chuỗi SMILES. Các descriptor này đại diện cho các thuộc tính lý hóa khác nhau của các hợp chất.)*
- **Python:** Store the calculated descriptors in the Pandas DataFrame alongside the activity data. *(Python: Lưu trữ các descriptor đã tính toán trong Pandas DataFrame cùng với dữ liệu hoạt tính.)*

Model Training and Evaluation (Huấn luyện và đánh giá mô hình):

- **Scikit-learn:** Use Scikit-learn to train and evaluate machine learning models. Choose appropriate models based on your prediction task (regression or classification). *(Scikit-learn: Sử dụng Scikit-learn để huấn luyện và đánh giá các mô hình học máy. Chọn các mô hình phù hợp dựa trên nhiệm vụ dự đoán của bạn (hồi quy hoặc phân loại).)*
- **Python:** Split the data into training and testing sets, train the model on the training set, and evaluate its performance on the testing set using appropriate metrics. *(Python: Chia dữ liệu thành tập huấn luyện và tập kiểm tra, huấn luyện mô hình trên tập huấn luyện và đánh giá hiệu suất của nó trên tập kiểm tra bằng cách sử dụng các metric phù hợp.)*

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

SQL (để trích xuất dữ liệu):

```sql
-- Lấy 100 dòng dữ liệu từ bảng 'activities' và 'molecule_dictionary'
SELECT md.chembl_id, md.smiles, act.standard_value, act.standard_type
FROM activities act
```

```sql
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0
LIMIT 100;
```

Lưu ý về lỗi SQL:

Lỗi `ERROR: operator does not exist: numeric ~ unknown` thường xảy ra khi bạn cố gắng sử dụng toán tử `~` (regular expression match) trên một cột số (numeric). Trong trường hợp này, bạn đang cố gắng so sánh `act.standard_value` (có thể là kiểu số) với một chuỗi regular expression `'^[0-9\.]+$'`.

Giải pháp:

Bạn có thể bỏ điều kiện `AND act.standard_value ~ '^[0-9\.]+$'` (nếu bạn đã đảm bảo rằng `standard_value` chỉ chứa số) hoặc chuyển đổi `standard_value` sang kiểu text trước khi so sánh:

```sql
-- Ví dụ chuyển đổi sang text trước khi so sánh (nếu cần)
SELECT md.chembl_id, md.smiles, act.standard_value, act.standard_type
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0
  AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$' -- Chuyển đổi sang TEXT
LIMIT 100;
```

Python (ví dụ cơ bản):

```python
import os
import pandas as pd
import psycopg2
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 numpy as np

# Cấu hình kết nối database
db_params = {
    'host': '192.168.206.136',
    'user': 'rd',
    'password': 'rd',
    'database': 'chembl_35'
}

# Hàm kết nối database và truy vấn dữ liệu
def fetch_data_from_chembl(query, db_params):
    conn = psycopg2.connect(**db_params)
    df = pd.read_sql_query(query, conn)
    conn.close()
    return df

# Hàm tính toán descriptor hóa học bằng RDKit
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        descriptors = {desc_name: desc_func(mol) for desc_name, desc_func in
```

```
Descriptors.descList}
        return descriptors
    else:
        return None

# Đường dẫn cơ sở của dự án
base_path = os.getcwd()  # Lấy thư mục hiện tại
data_path = os.path.join(base_path, 'data')
notebook_path = os.path.join(base_path, 'notebooks')

# Tạo thư mục nếu chưa tồn tại
os.makedirs(data_path, exist_ok=True)
os.makedirs(notebook_path, exist_ok=True)

# Truy vấn SQL để lấy dữ liệu
sql_query = """
SELECT md.chembl_id, md.smiles, act.standard_value, act.standard_type
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0
LIMIT 100;
"""

# Lấy dữ liệu từ ChEMBL
data = fetch_data_from_chembl(sql_query, db_params)

# Tiền xử lý dữ liệu
data = data.dropna(subset=['smiles', 'standard_value'])  # Loại bỏ các hàng có giá trị
NaN trong cột 'smiles' hoặc 'standard_value'

# Chuyển đổi IC50 sang pIC50
data['pIC50'] = -np.log10(data['standard_value'] / 1e9) #standard_value in nM

# Tính toán descriptor hóa học
data['descriptors'] = data['smiles'].apply(calculate_descriptors)

# Loại bỏ các hàng có lỗi tính toán descriptor
data = data.dropna(subset=['descriptors'])

# Chuyển đổi descriptor thành DataFrame riêng
descriptors_df = pd.DataFrame(data['descriptors'].tolist(), index=data.index)

# Gộp descriptor vào DataFrame chính
data = pd.concat([data, descriptors_df], axis=1)

# Loại bỏ cột 'descriptors' và 'smiles' không cần thiết
data = data.drop(columns=['descriptors', 'smiles'])

# Chọn các features và target
X = data.drop(columns=['chembl_id', 'standard_value', 'standard_type', 'pIC50'])
y = data['pIC50']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```python
# Huấn luyện mô hình Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
y_pred = model.predict(X_test)

# Đánh giá mô hình
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Lưu mô hình (tùy chọn)
# import joblib
# model_filename = os.path.join(base_path, 'model.pkl')
# joblib.dump(model, model_filename)

# Lưu dữ liệu vào file CSV (tùy chọn)
csv_filename = os.path.join(data_path, 'chembl_data.csv')
data.to_csv(csv_filename, index=False)

print(f"Data saved to {csv_filename}")
```

Lưu ý:

- Đảm bảo rằng bạn đã cài đặt các thư viện cần thiết (RDKit, Pandas, Scikit-learn, psycopg2).
- Điều chỉnh `db_params` với thông tin kết nối cơ sở dữ liệu của bạn.
- Code này chỉ là một ví dụ cơ bản. Bạn có thể cần điều chỉnh nó để phù hợp với nhu cầu cụ thể của dự án của bạn.
- Việc chọn các descriptor phù hợp và thuật toán machine learning tốt nhất đòi hỏi thử nghiệm và đánh giá kỹ lưỡng.

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

SQL Examples:

1. **Lấy số lượng hợp chất cho mỗi standard_type:**

```sql
SELECT standard_type, COUNT(*) AS compound_count
FROM activities
GROUP BY standard_type
ORDER BY compound_count DESC
LIMIT 10;
```

2. **Lấy các hợp chất có khối lượng phân tử (molecular weight) nằm trong một khoảng nhất định (sử dụng dữ liệu đã được tính toán và lưu trữ trong bảng khác):**

```sql
-- Giả sử bạn có bảng 'molecular_properties' với cột 'molregno' và 'molecular_weight'
SELECT md.chembl_id, mp.molecular_weight
FROM molecule_dictionary md
JOIN molecular_properties mp ON md.molregno = mp.molregno
WHERE mp.molecular_weight BETWEEN 200 AND 500
LIMIT 10;
```

3. **Lấy thông tin về các hợp chất hoạt động trên một mục tiêu (target) cụ thể:**

```sql
SELECT md.chembl_id, act.standard_value, act.standard_type
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.target_chembl_id = 'CHEMBL205' -- Thay thế bằng target_chembl_id mong muốn
  AND act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
LIMIT 10;
```

4. **Lấy các hợp chất có chứa một motif (phân tử con) cụ thể (sử dụng SMILES và LIKE):**

```sql
SELECT chembl_id, smiles
FROM molecule_dictionary
WHERE smiles LIKE '%C(=O)N%' -- Ví dụ: tìm các hợp chất chứa amide bond
LIMIT 10;
```

5. **Lấy các hợp chất có hoạt tính tốt nhất (IC50 thấp nhất) trên một target cụ thể:**

```sql
SELECT md.chembl_id, act.standard_value
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.target_chembl_id = 'CHEMBL205'
  AND act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
ORDER BY act.standard_value ASC
LIMIT 10;
```

Python Examples:

1. **Tính toán LogP bằng RDKit:**

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

def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        return Descriptors.MolLogP(mol)
    else:
        return None

# Sử dụng hàm này trên cột 'smiles' của DataFrame:
# data['logp'] = data['smiles'].apply(calculate_logp)
```

2. **Tính toán TPSA (Topological Polar Surface Area):**

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

def calculate_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        return Descriptors.TPSA(mol)
    else:
        return None

# Sử dụng hàm này trên cột 'smiles' của DataFrame:
# data['tpsa'] = data['smiles'].apply(calculate_tpsa)
```

3. **Chuẩn hóa SMILES (SMILES Standardization):**

```python
from rdkit import Chem
from rdkit.Chem import MolStandardize

def standardize_smiles(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        # Tạo một phiên bản của Standardizer
        std = MolStandardize.Standardizer()
        mol = std.charge_parent(mol)
        return Chem.MolToSmiles(mol)
    else:
        return None
```

```
# Sử dụng hàm này trên cột 'smiles' của DataFrame:
# data['standardized_smiles'] = data['smiles'].apply(standardize_smiles)
```

4. **Sử dụng Random Forest Regression:**

```
from sklearn.ensemble import RandomForestRegressor

# Khởi tạo và huấn luyện mô hình
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Dự đoán và đánh giá
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
print(f'Random Forest MSE: {mse_rf}')
```

5. **Sử dụng Gradient Boosting Regression (ví dụ XGBoost):**

```
import xgboost as xgb

# Khởi tạo và huấn luyện mô hình
xgb_model = xgb.XGBRegressor(n_estimators=100, random_state=42)
xgb_model.fit(X_train, y_train)

# Dự đoán và đánh giá
y_pred_xgb = xgb_model.predict(X_test)
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
print(f'XGBoost MSE: {mse_xgb}')
```

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