PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Trích xuất hợp chất gắn kinase

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code mẫu và các ví dụ, được thiết kế để giúp bạn triển khai dự án một cách hiệu quả nhất theo chuẩn AIMLOps Template.

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

Mục tiêu:

- **Xây dựng tập dữ liệu:** Tạo một tập dữ liệu từ ChEMBL 35 chứa thông tin về cấu trúc hóa học (SMILES), hoạt tính sinh học (IC50, Ki, v.v.) và các thuộc tính liên quan khác.
- **Tính toán Descriptors:** Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc SMILES. Các descriptor này đại diện cho các đặc điểm hóa lý của phân tử.
- **Phân tích Tương quan:** Phân tích mối tương quan giữa các descriptor và hoạt tính sinh học. Điều này giúp xác định các đặc điểm cấu trúc quan trọng ảnh hưởng đến hoạt tính.
- **Xây dựng Mô hình Dự đoán:** Xây dựng mô hình học máy để dự đoán hoạt tính sinh học dựa trên các descriptor.
- **Đánh giá Mô hình:** Đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra.

Các Bước Chính:

1. **Kết nối đến ChEMBL Database:** Sử dụng thông tin kết nối bạn cung cấp để truy cập dữ liệu.
2. **Truy vấn Dữ liệu:** Sử dụng SQL để truy vấn dữ liệu từ các bảng liên quan trong ChEMBL, như `activities`, `molecule_dictionary`, `compound_structures`, v.v.
3. **Tiền xử lý Dữ liệu:** Làm sạch và chuẩn hóa dữ liệu, loại bỏ các giá trị ngoại lệ hoặc thiếu.
4. **Tính toán Descriptors:** Sử dụng RDKit để chuyển đổi SMILES thành các đối tượng phân tử và tính toán các descriptor.
5. **Phân tích Tương quan:** Sử dụng các phương pháp thống kê để xác định mối tương quan giữa các descriptor và hoạt tính sinh học.
6. **Xây dựng Mô hình:** Chọn một thuật toán học máy phù hợp (ví dụ: Random Forest, SVM, Linear Regression) và huấn luyện mô hình trên tập dữ liệu huấn luyện.
7. **Đánh giá Mô hình:** Đánh giá mô hình trên tập dữ liệu kiểm tra bằng các chỉ số như R-squared, RMSE, MAE.

2. Hướng Dẫn Song Ngữ (Bilingual Guidance)

Bước 1: Kết nối đến ChEMBL Database (Connecting to ChEMBL Database)

- **Tiếng Việt:** Sử dụng thông tin kết nối (IP, user, pass, database) để kết nối đến cơ sở dữ liệu ChEMBL 35 trên PostgreSQL.
- **English:** Use the connection information (IP, user, pass, database) to connect to the ChEMBL 35 database on PostgreSQL.

Bước 2: Truy vấn Dữ liệu (Data Query)

- **Tiếng Việt:** Sử dụng SQL để truy vấn dữ liệu hoạt tính sinh học, cấu trúc SMILES và các thông tin liên quan khác từ cơ sở dữ liệu ChEMBL.
- **English:** Use SQL to query bioactivity data, SMILES structures, and other relevant information from the ChEMBL database.

Bước 3: Tiền Xử Lý Dữ Liệu (Data Preprocessing)

- **Tiếng Việt:** Làm sạch dữ liệu, xử lý các giá trị thiếu, và chuẩn hóa dữ liệu hoạt tính sinh học (ví dụ: chuyển đổi IC50 sang pIC50).
- **English:** Clean the data, handle missing values, and standardize bioactivity data (e.g., convert IC50 to pIC50).

Bước 4: Tính Toán Descriptors (Descriptor Calculation)

- **Tiếng Việt:** Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc SMILES.
- **English:** Use RDKit to calculate molecular descriptors from SMILES structures.

Bước 5: Phân Tích Tương Quan (Correlation Analysis)

- **Tiếng Việt:** Phân tích mối tương quan giữa các descriptor và hoạt tính sinh học để xác định các yếu tố quan trọng.
- **English:** Analyze the correlation between descriptors and bioactivity to identify important factors.

Bước 6: Xây Dựng Mô Hình (Model Building)

- **Tiếng Việt:** Xây dựng mô hình học máy để dự đoán hoạt tính sinh học dựa trên các descriptor.
- **English:** Build a machine learning model to predict bioactivity based on descriptors.

Bước 7: Đánh Giá Mô Hình (Model Evaluation)

- **Tiếng Việt:** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số đánh giá phù hợp.
- **English:** Evaluate the performance of the model using appropriate evaluation metrics.

3. Code SQL, Python (English)

SQL (to extract 100 rows of data):

```sql
-- SQL query to extract 100 rows of data
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND cs.canonical_smiles IS NOT NULL
    AND act.standard_value ~ '^[0-9\.]+$' -- fix error a
LIMIT 100;
```

Python (Jupyter Notebook):

```python
import os
import pandas as pd
import psycopg2
from rdkit import Chem
from rdkit.Chem import Descriptors
```

```
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
from sklearn.preprocessing import StandardScaler

# Database connection details
ip = '192.168.206.136'
user = 'rd'
password = 'rd'
database = 'chembl_35'

# Base path for AIMLOps template
base_path = '../data'  # Adjust if needed

# SQL query to extract data
sql_query = """
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND cs.canonical_smiles IS NOT NULL
    AND act.standard_value ~ '^[0-9\.]+$'
LIMIT 100;
"""

# Function to connect to the database and execute the query
def fetch_data(ip, user, password, database, query):
    conn = None
    try:
        conn = psycopg2.connect(host=ip, user=user, password=password,
database=database)
        df = pd.read_sql_query(query, conn)
        return df
    except psycopg2.Error as e:
        print(f"Error connecting to the database: {e}")
        return None
    finally:
        if conn:
            conn.close()

# Fetch data from the database
df = fetch_data(ip, user, password, database, sql_query)

# Check if data was fetched successfully
if df is None or df.empty:
    print("Failed to fetch data from the database.")
    exit()
```

```
# Data preprocessing: Convert IC50 to pIC50
df = df.dropna(subset=['standard_value', 'canonical_smiles'])
df = df[df['standard_value'].astype(str).str.match(r'^[0-9\.]+$')]  # Keep only
numeric values
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df = df.dropna(subset=['standard_value'])
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Convert nM to M and then to
pIC50

# RDKit: Calculate molecular descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    for name, func in Descriptors.descList:
        try:
            descriptors[name] = func(mol)
        except:
            descriptors[name] = None
    return descriptors

df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors'])

# Convert descriptors to DataFrame
descriptors_df = pd.DataFrame(df['descriptors'].tolist())
df = pd.concat([df, descriptors_df], axis=1)

# Prepare data for machine learning
X = df.drop(['chembl_id', 'canonical_smiles', 'standard_type', 'standard_value',
'standard_units', 'pIC50', 'descriptors'], axis=1)
y = df['pIC50']

# Handle missing values by filling with the mean
X = X.fillna(X.mean())

# Scale the features
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 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}")
```

```
# Save results to CSV
output_file = os.path.join(base_path, 'chembl_35_results.csv')
df.to_csv(output_file, index=False)
print(f"Results saved to {output_file}")
```

Giải thích Code:

1. **Kết nối Database:** Sử dụng `psycopg2` để kết nối đến database ChEMBL 35.
2. **Truy Vấn SQL:** Truy vấn dữ liệu cần thiết từ các bảng `activities`, `molecule_dictionary`, và `compound_structures`.
3. **Tiền Xử Lý:**
    - Chuyển đổi giá trị IC50 sang pIC50.
    - Loại bỏ các giá trị thiếu.
4. **Tính Toán Descriptors:**
    - Sử dụng RDKit để chuyển đổi SMILES thành các descriptor phân tử.
5. **Xây Dựng Mô Hình:**
    - Sử dụng `RandomForestRegressor` từ `sklearn`.
    - Chia dữ liệu thành tập huấn luyện và tập kiểm tra.
    - Huấn luyện mô hình và đánh giá hiệu suất.
6. **Lưu Kết Quả:** Lưu kết quả vào file CSV.

Lưu ý:

- Đảm bảo đã cài đặt các thư viện cần thiết: `psycopg2`, `rdkit`, `pandas`, `scikit-learn`.
- Điều chỉnh đường dẫn `base_path` cho phù hợp với cấu trúc thư mục của bạn.
- Phiên bản scikit-learn cũ có thể không hỗ trợ `squared=False` trong `mean_squared_error`. Bạn có thể bỏ tham số này nếu gặp lỗi.
- Sửa lỗi "ERROR: operator does not exist: numeric ~ unknown" bằng cách thêm dòng `AND act.standard_value ~ '^[0-9\.]+$'` để đảm bảo chỉ lấy các giá trị số.

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

Ví dụ 1: Lọc dữ liệu theo khoảng giá trị IC50 (Filtering data by IC50 range)

- **SQL:**

```
SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value BETWEEN 100 AND 1000 -- IC50 between 100 and 1000 nM
    AND act.standard_value ~ '^[0-9\.]+$'
LIMIT 100;
```

- **Python:**

```
# Add this condition to the SQL query in the Python code
sql_query = """
SELECT
```

```
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value BETWEEN 100 AND 1000
    AND act.standard_value ~ '^[0-9\.]+$'
LIMIT 100;
"""
```

Ví dụ 2: Tính toán số lượng các chất có cùng scaffold (Calculating the number of compounds with the same scaffold)

- **SQL:**

```
-- Requires a table with pre-calculated scaffolds
-- Assuming you have a 'scaffolds' table with 'molregno' and 'murcko_scaffold' columns
SELECT
    s.murcko_scaffold,
    COUNT(DISTINCT md.chembl_id) AS compound_count
FROM
    scaffolds s
JOIN
    molecule_dictionary md ON s.molregno = md.molregno
GROUP BY
    s.murcko_scaffold
ORDER BY
    compound_count DESC
LIMIT 10;
```

- **Python:**

```
from rdkit.Chem import MurckoScaffold

def calculate_murcko_scaffold(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return MurckoScaffold.GetScaffoldForMol(mol).GetSmiles()
    return None

df['murcko_scaffold'] = df['canonical_smiles'].apply(calculate_murcko_scaffold)
scaffold_counts = df['murcko_scaffold'].value_counts().reset_index()
scaffold_counts.columns = ['murcko_scaffold', 'compound_count']
print(scaffold_counts.head(10))
```

Ví dụ 3: Sử dụng các loại descriptors khác nhau (Using different types of descriptors)

- **Python:**

```
from rdkit.Chem import Crippen, Lipinski

def calculate_more_descriptors(mol):
    if mol is None:
        return {}
```

```
    descriptors = {
        "MolLogP": Crippen.MolLogP(mol),
        "NumHAcceptors": Lipinski.NumHAcceptors(mol),
        "NumHDonors": Lipinski.NumHDonors(mol)
    }
    return descriptors

df['additional_descriptors'] = df['canonical_smiles'].apply(lambda x:
calculate_more_descriptors(Chem.MolFromSmiles(x)))
additional_descriptors_df = pd.DataFrame(df['additional_descriptors'].tolist())
df = pd.concat([df, additional_descriptors_df], axis=1)
```

Ví dụ 4: Thay đổi thuật toán học máy (Changing the machine learning algorithm)

- **Python:**

```
from sklearn.svm import SVR

# Train a Support Vector Regression model
model = SVR(kernel='rbf')  # You can also try 'linear', 'poly', 'sigmoid'
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}")
```

Ví dụ 5: Lưu mô hình đã huấn luyện (Saving the trained model)

- **Python:**

```
import joblib

# Train your model...

# Save the model
model_file = os.path.join(base_path, 'chembl_35_model.pkl')
joblib.dump(model, model_file)
print(f"Model saved to {model_file}")

# Load the model later
loaded_model = joblib.load(model_file)
```

Hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn triển khai dự án phân tích dữ liệu ChEMBL 35 một cách hiệu quả. Chúc bạn thành công!