PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Liên kết dữ liệu thuốc với các pha lâm sàng

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Với vai trò là một chuyên gia nghiên cứu và phát triển dược học, tôi hiểu rõ tầm quan trọng của việc khai thác dữ liệu ChEMBL để tìm kiếm các hợp chất tiềm năng.

Dưới đây là phân tích chi tiết, hướng dẫn song ngữ, code SQL và Python mẫu, cùng với các ví dụ để giúp bạn bắt đầu dự án của mình.

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

- **Mục tiêu:**
    - Xây dựng mô hình dự đoán hoạt tính của các hợp chất dựa trên cấu trúc hóa học của chúng.
    - Xác định các đặc tính hóa học quan trọng ảnh hưởng đến hoạt tính.
    - Tìm kiếm các hợp chất tiềm năng có hoạt tính mong muốn.
- **Dữ liệu:**
    - Dữ liệu từ ChEMBL 35, bao gồm thông tin về cấu trúc hóa học (SMILES), hoạt tính sinh học (IC50, Ki, EC50, v.v.), và các thuộc tính khác.
    - Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc SMILES.
- **Mô hình:**
    - **Hồi quy (Regression):** Dự đoán giá trị hoạt tính liên tục (ví dụ: pIC50).
        - **Ví dụ:** Hồi quy tuyến tính (Linear Regression), Hồi quy Ridge (Ridge Regression), Hồi quy Lasso (Lasso Regression), Máy vector hỗ trợ hồi quy (Support Vector Regression - SVR), Rừng ngẫu nhiên (Random Forest), Gradient Boosting.
    - **Phân loại (Classification):** Dự đoán một hợp chất có hoạt tính hay không (ví dụ: hoạt tính > ngưỡng).
        - **Ví dụ:** Hồi quy Logistic (Logistic Regression), Máy vector hỗ trợ phân loại (Support Vector Classification - SVC), Cây quyết định (Decision Tree), Rừng ngẫu nhiên (Random Forest), Gradient Boosting.
- **Quy trình:**
    1. **Thu thập dữ liệu (Data Acquisition):** Truy vấn cơ sở dữ liệu ChEMBL để lấy dữ liệu cần thiết.
    2. **Tiền xử lý dữ liệu (Data Preprocessing):**
        - Làm sạch dữ liệu (xử lý giá trị thiếu, loại bỏ dữ liệu trùng lặp).
        - Chuyển đổi dữ liệu (ví dụ: chuyển IC50 sang pIC50).
    3. **Tính toán descriptor phân tử (Molecular Descriptor Calculation):** Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc SMILES.
    4. **Lựa chọn đặc trưng (Feature Selection):** Chọn các descriptor quan trọng nhất để đưa vào mô hình.
    5. **Xây dựng mô hình (Model Building):** Chọn và huấn luyện mô hình phù hợp.
    6. **Đánh giá mô hình (Model Evaluation):** Đánh giá hiệu suất của mô hình bằng các độ đo phù hợp (ví dụ: R-squared, RMSE, AUC).
    7. **Dự đoán và phân tích (Prediction and Analysis):** Sử dụng mô hình để dự đoán hoạt tính của các hợp chất mới và phân tích các đặc tính quan trọng.

2. Hướng dẫn Song ngữ (Bilingual Guide)

Bước	Tiếng Anh	Tiếng Việt
1. Thu thập dữ liệu	Data Acquisition: Query the ChEMBL database to retrieve the necessary data.	Thu thập dữ liệu: Truy vấn cơ sở dữ liệu ChEMBL để lấy dữ liệu cần thiết.
2. Tiền xử lý dữ liệu	Data Preprocessing: Clean the data (handle missing values, remove duplicates). Transform the data (e.g., convert IC50 to pIC50).	Tiền xử lý dữ liệu: Làm sạch dữ liệu (xử lý giá trị thiếu, loại bỏ dữ liệu trùng lặp). Chuyển đổi dữ liệu (ví dụ: chuyển IC50 sang pIC50).
3. Tính toán descriptor phân tử	Molecular Descriptor Calculation: Use RDKit to calculate molecular descriptors from SMILES structures.	Tính toán descriptor phân tử: Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc SMILES.
4. Lựa chọn đặc trưng	Feature Selection: Select the most important descriptors to include in the model.	Lựa chọn đặc trưng: Chọn các descriptor quan trọng nhất để đưa vào mô hình.
5. Xây dựng mô hình	Model Building: Select and train an appropriate model.	Xây dựng mô hình: Chọn và huấn luyện mô hình phù hợp.
6. Đánh giá mô hình	Model Evaluation: Evaluate the model's performance using appropriate metrics (e.g., R-squared, RMSE, AUC).	Đánh giá mô hình: Đánh giá hiệu suất của mô hình bằng các độ đo phù hợp (ví dụ: R-squared, RMSE, AUC).
7. Dự đoán và phân tích	Prediction and Analysis: Use the model to predict the activity of new compounds and analyze important characteristics.	Dự đoán và phân tích: Sử dụng mô hình để dự đoán hoạt tính của các hợp chất mới và phân tích các đặc tính quan trọng.
Lỗi thường gặp (Common Errors)	ERROR: operator does not exist: numeric ~ unknown: This error occurs because you're trying to use the ~ operator (regular expression matching) on a numeric column. You need to cast the column to text first.	Lỗi: operator does not exist: numeric ~ unknown: Lỗi này xảy ra khi bạn cố gắng sử dụng toán tử ~ (so khớp biểu thức chính quy) trên một cột số. Bạn cần chuyển đổi cột thành văn bản trước.
Scikit-learn versi	Older scikit-learn versions may not support `squared=False` in `mean_squared_error`. Update your scikit-learn version or remove the argument.	Phiên bản Scikit-learn cũ có thể không hỗ trợ `squared=False` trong `mean_squared_error`. Hãy cập nhật phiên bản Scikit-learn của bạn hoặc xóa đối số này.

Bước	Tiếng Anh	Tiếng Việt
	on issue	

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

SQL (Ví dụ: Lấy dữ liệu từ ChEMBL cho một target cụ thể):

```sql
-- Get data for a specific target (e.g., target_chembl_id = 'CHEMBL205')
SELECT
    cmp.chembl_id,
    cmp.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.target_chembl_id = 'CHEMBL205'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$'  -- Corrected line: Casting to text for regex
LIMIT 100;

--Giải thích:
-- Lấy thông tin về hợp chất, cấu trúc, loại hoạt tính, giá trị, và đơn vị
-- Từ bảng cấu trúc hợp chất, hoạt động, và từ điển mục tiêu
-- Lọc theo target_chembl_id cụ thể (ví dụ: CHEMBL205), loại hoạt tính là IC50, đơn vị là nM, giá trị không null, và giá trị chỉ chứa số và dấu chấm (để đảm bảo tính hợp lệ)
-- Giới hạn kết quả 100 dòng

-- Explanation:
-- Get compound information, structure, activity type, value, and units
-- From the compound structure, activities, and target dictionary tables
-- Filter by specific target_chembl_id (e.g., CHEMBL205), activity type is IC50, units are nM, value is not null, and value contains only numbers and dots (to ensure validity)
-- Limit the result to 100 rows
```

Python (Ví dụ: Tính toán descriptor phân tử và xây dựng mô hình):

```python
import os
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
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# Đường dẫn cơ sở (Base path)
```

```
base_path = "../data"

# Tên file CSV (CSV file name)
csv_file = "chembl_205_ic50_100.csv"  # Replace with your actual file name
file_path = os.path.join(base_path, csv_file)

# Đọc dữ liệu từ file CSV (Read data from CSV file)
try:
    df = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"Error: File not found at {file_path}")
    exit()

# In ra thông tin cơ bản của DataFrame (Print basic DataFrame info)
print("DataFrame Info:")
print(df.info())

# Hàm tính toán descriptor phân tử (Function to 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] = np.nan  # Handle potential errors
    return descriptors

# Áp dụng hàm tính toán descriptor (Apply descriptor calculation function)
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)

# Loại bỏ các dòng có lỗi (Remove rows with errors)
df = df.dropna(subset=['descriptors'])

# Chuyển đổi descriptor thành DataFrame (Convert descriptors to DataFrame)
descriptors_df = pd.DataFrame(df['descriptors'].tolist())

# Gộp DataFrame descriptor với DataFrame chính (Merge descriptor DataFrame with main
DataFrame)
df = pd.concat([df, descriptors_df], axis=1)

# Chuyển đổi IC50 thành pIC50 (Convert IC50 to pIC50)
df['pIC50'] = -np.log10(df['standard_value'] * 1e-9)  # Convert nM to M

# Chọn các đặc trưng và biến mục tiêu (Select features and target variable)
features = descriptors_df.columns
target = 'pIC50'

# Loại bỏ các giá trị vô cực và NaN (Remove infinite and NaN values)
df = df[np.isfinite(df[target])]
df = df.dropna(axis=1, how='any')

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra (Split data into training and test
sets)
X_train, X_test, y_train, y_test = train_test_split(df[features].fillna(0),
df[target], test_size=0.2, random_state=42)
```

```python
# Chuẩn hóa dữ liệu và xây dựng mô hình hồi quy tuyến tính (Standardize data and build linear regression model)
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LinearRegression())
])

# Huấn luyện mô hình (Train the model)
pipeline.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra (Predict on the test set)
y_pred = pipeline.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}')

# In ra 10 dòng đầu của DataFrame (Print the first 10 rows of the DataFrame)
print("First 10 rows of DataFrame:")
print(df.head(10))
```

Giải thích:

1. **Import thư viện:** Import các thư viện cần thiết (pandas, RDKit, scikit-learn).
2. **Đọc dữ liệu:** Đọc dữ liệu từ file CSV.
3. **Tính toán descriptor:** Tính toán descriptor phân tử bằng RDKit.
4. **Tiền xử lý dữ liệu:** Xử lý giá trị thiếu, chuyển đổi IC50 sang pIC50.
5. **Chia dữ liệu:** Chia dữ liệu thành tập huấn luyện và tập kiểm tra.
6. **Xây dựng mô hình:** Xây dựng mô hình hồi quy tuyến tính.
7. **Đánh giá mô hình:** Đánh giá hiệu suất của mô hình.

Explanation:

1. **Import libraries:** Import necessary libraries (pandas, RDKit, scikit-learn).
2. **Read data:** Read data from CSV file.
3. **Calculate descriptors:** Calculate molecular descriptors using RDKit.
4. **Data preprocessing:** Handle missing values, convert IC50 to pIC50.
5. **Split data:** Split data into training and test sets.
6. **Build model:** Build a linear regression model.
7. **Evaluate model:** Evaluate the model's performance.

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

Ví dụ 1: Lọc các hợp chất có hoạt tính IC50 dưới 100 nM (SQL)

```sql
SELECT
    cmp.chembl_id,
    cmp.canonical_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
```

```
    td.target_chembl_id = 'CHEMBL205'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value <= 100
LIMIT 100;
```

Ví dụ 1: Tính toán logP và MW (Python)

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

def calculate_logp_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None, None
    logp = Descriptors.MolLogP(mol)
    mw = Descriptors.MolWt(mol)
    return logp, mw

smiles = 'CC(=O)Oc1ccccc1C(=O)O'
logp, mw = calculate_logp_mw(smiles)
print(f'LogP: {logp}, MW: {mw}')
```

Ví dụ 2: Lấy các hợp chất tác động lên một protein cụ thể (SQL)

```
SELECT
    cmp.chembl_id,
    cmp.canonical_smiles
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.target_chembl_id = 'CHEMBL301' -- Replace with desired target
LIMIT 100;
```

Ví dụ 2: Tính toán TPSA (Python)

```
from rdkit import Chem
from rdkit.Chem import rdMolDescriptors

def calculate_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    tpsa = rdMolDescriptors.CalcTPSA(mol)
    return tpsa

smiles = 'c1ccccc1O'
tpsa = calculate_tpsa(smiles)
print(f'TPSA: {tpsa}')
```

Ví dụ 3: Tìm các hợp chất có Ki < 10 nM (SQL)

```
SELECT
    cmp.chembl_id,
    cmp.canonical_smiles,
    act.standard_value
FROM
    compound_structures cmp
```

```sql
JOIN
    activities act ON cmp.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.target_chembl_id = 'CHEMBL205'
    AND act.standard_type = 'Ki'
    AND act.standard_units = 'nM'
    AND act.standard_value < 10
LIMIT 100;
```

Ví dụ 3: Tính toán số lượng vòng (Python)

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

def calculate_ring_count(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    ring_count = Descriptors.RingCount(mol)
    return ring_count

smiles = 'c1ccccc1Cc2ccccc2'
ring_count = calculate_ring_count(smiles)
print(f'Ring Count: {ring_count}')
```

Ví dụ 4: Lấy các hợp chất có hoạt tính trên một đường dẫn sinh học cụ thể (SQL)

```sql
SELECT
    cmp.chembl_id,
    cmp.canonical_smiles
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
JOIN
    target_components tc ON td.tid = tc.tid
JOIN
    component_sequences cs ON tc.component_id = cs.component_id
WHERE
    cs.accession = 'P00533'  -- Example: EGFR
LIMIT 100;
```

Ví dụ 4: Scale dữ liệu sử dụng StandardScaler (Python)

```python
from sklearn.preprocessing import StandardScaler
import numpy as np

# Ví dụ dữ liệu
data = np.array([[1.0], [2.0], [3.0], [4.0], [5.0]])

# Khởi tạo StandardScaler
scaler = StandardScaler()

# Fit và transform dữ liệu
scaled_data = scaler.fit_transform(data)
```

```
print("Original data:\n", data)
print("Scaled data:\n", scaled_data)
```

Ví dụ 5: Lấy dữ liệu pIC50 từ ChEMBL (SQL)

```
SELECT
    cmp.chembl_id,
    cmp.canonical_smiles,
    -LOG10(act.standard_value * POW(10, -9)) AS pIC50 -- Calculate pIC50
FROM
    compound_structures cmp
JOIN
    activities act ON cmp.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.target_chembl_id = 'CHEMBL205'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$' -- Ensure the value is numeric
LIMIT 100;
```

Ví dụ 5: Sử dụng Random Forest Regressor (Python)

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

# Tạo dữ liệu ví dụ
data = {'feature1': np.random.rand(100),
        'feature2': np.random.rand(100),
        'target': 2 * np.random.rand(100)}
df = pd.DataFrame(data)

# Chia dữ liệu thành features (X) và target (y)
X = df[['feature1', 'feature2']]
y = df['target']

# Chia dữ liệu thành training set và test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Khởi tạo Random Forest Regressor
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

# Huấn luyện mô hình
rf_regressor.fit(X_train, y_train)

# Dự đoán trên test set
y_pred = rf_regressor.predict(X_test)

# Đánh giá mô hình
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

Lưu ý quan trọng:

- **Thay đổi đường dẫn:** Thay đổi `base_path` và `csv_file` cho phù hợp với cấu trúc thư mục và tên file của bạn.
- **Xử lý lỗi:** Các đoạn code trên chỉ là ví dụ, bạn cần thêm các xử lý lỗi để đảm bảo chương trình chạy ổn định.
- **Lựa chọn mô hình:** Lựa chọn mô hình phù hợp với dữ liệu và mục tiêu của bạn.
- **Tối ưu hóa mô hình:** Tối ưu hóa các tham số của mô hình để đạt được hiệu suất tốt nhất.
- **Phiên bản scikit-learn:** Hãy đảm bảo bạn đang sử dụng phiên bản scikit-learn tương thích. Nếu bạn gặp lỗi với tham số `squared=False` trong `mean_squared_error`, hãy cập nhật scikit-learn hoặc bỏ tham số đó.
- **Kiểm tra kiểu dữ liệu:** Đảm bảo kiểu dữ liệu của các cột phù hợp với các phép toán và hàm bạn sử dụng.

Hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án phân tích dữ liệu ChEMBL 35 của mình một cách hiệu quả. Nếu bạn có bất kỳ câu hỏi nào, đừng ngần ngại hỏi!