# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 💉



## 🧩 Topic: Ưu tiên hợp chất cho quá trình tổng hợp

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 R&D dược học, tôi hiểu rõ tầm quan trọng của việc khai thác dữ liệu lớn để đẩy nhanh quá trình phát triển thuốc. Chúng ta hãy cùng nhau xây dựng một quy trình làm việc hiệu quả và chuẩn chỉnh.

**Tổng quan về dự án:**

Dự án của bạn tập trung vào việc phân tích dữ liệu từ cơ sở dữ liệu ChEMBL 35, một nguồn tài nguyên vô giá chứa thông tin về các phân tử sinh học hoạt tính. Sử dụng RDKit, một toolkit mạnh mẽ cho tin học hóa học và tin sinh học, bạn có thể trích xuất các đặc trưng hóa học, tính toán các descriptor phân tử và xây dựng các mô hình dự đoán hoạt tính sinh học.

**Cấu trúc thư mục AIMLOps:**

Việc tuân thủ cấu trúc thư mục AIMLOps là một cách tiếp cận tốt để đảm bảo tính tái sử dụng, khả năng mở rộng và dễ bảo trì của dự án. Bạn đã đề cập đến việc sử dụng `os.path.join(base_path, ...)` để quản lý đường dẫn, đây là một thói quen tốt.

**Quy trình làm việc:**

1. **Trích xuất dữ liệu từ ChEMBL 35 (SQL):** Sử dụng PostgreSQL (psql) để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35. Lưu kết quả vào các file CSV trong thư mục `../data/`.
2. **Phân tích dữ liệu và xây dựng mô hình (Python/Jupyter Notebook):** Sử dụng Jupyter Notebook để đọc dữ liệu CSV, tiền xử lý dữ liệu, trích xuất đặc trưng bằng RDKit và xây dựng các mô hình học máy. Lưu các notebook với tên `Topic_CheMBL_35_14_1_*` và `Topic_CheMBL_35_14_2_*`.

**Phân tích, hướng dẫn song ngữ và code mẫu:**

Dưới đây là một số ví dụ về các phân tích bạn có thể thực hiện, cùng với hướng dẫn song ngữ và code mẫu (SQL và Python). Tôi sẽ tập trung vào việc giải thích các khái niệm và cung cấp các ví dụ code đơn giản, dễ hiểu.

**1. Phân tích phân bố hoạt tính (Activity Distribution Analysis):**

- **Mô tả:** Phân tích sự phân bố của các giá trị hoạt tính (ví dụ: IC50, Ki) để hiểu rõ hơn về phạm vi hoạt tính của các hợp chất trong tập dữ liệu.
- **SQL (Tiếng Anh):**

```sql
-- Select target name and standard relation, standard value.
SELECT
    target.pref_name,
    act.standard_relation,
    act.standard_value
FROM
    activities act
JOIN
    target_dictionary target ON act.tid = target.tid
WHERE
    target.target_type = 'SINGLE PROTEIN'
```

```
    AND act.standard_type = 'IC50'
LIMIT 100;
```

- **Giải thích:** Đoạn SQL này truy vấn cơ sở dữ liệu ChEMBL để lấy thông tin về tên mục tiêu (protein), loại quan hệ (ví dụ: "="), và giá trị hoạt tính (ví dụ: IC50) cho các hoạt chất tác động lên một protein đơn lẻ. Lệnh `LIMIT 100` giới hạn kết quả trả về 100 dòng.
- **Explanation:** This SQL query retrieves target name (protein), standard relation (e.g., "="), and activity value (e.g., IC50) for compounds acting on a single protein from the ChEMBL database. `LIMIT 100` restricts the result to 100 rows.

- **Python:**

```
import pandas as pd
import matplotlib.pyplot as plt

# Load data from CSV
data = pd.read_csv('your_data.csv')

# Filter data based on standard relation (e.g., '=')
data = data[data['standard_relation'] == '=']

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

# Remove rows with NaN in 'standard_value'
data = data.dropna(subset=['standard_value'])

# Plot histogram of activity values
plt.hist(data['standard_value'], bins=50)
plt.xlabel('IC50 Value')
plt.ylabel('Frequency')
plt.title('Distribution of IC50 Values')
plt.show()
```

- **Giải thích:** Đoạn Python này đọc dữ liệu từ file CSV, lọc các giá trị hoạt tính có quan hệ "=" (ví dụ: IC50 = x nM), chuyển đổi cột 'standard_value' sang kiểu số, loại bỏ các giá trị NaN và vẽ biểu đồ histogram để hiển thị sự phân bố của các giá trị IC50.
- **Explanation:** This Python script reads data from a CSV file, filters activity values with "=" relation (e.g., IC50 = x nM), converts the 'standard_value' column to numeric type, removes NaN values, and plots a histogram to visualize the distribution of IC50 values.

**2. Trích xuất SMILES và tính toán descriptor phân tử (SMILES Extraction and Molecular Descriptor Calculation):**

- **Mô tả:** Trích xuất chuỗi SMILES (Simplified Molecular Input Line Entry System) từ cơ sở dữ liệu và sử dụng RDKit để tính toán các descriptor phân tử như trọng lượng phân tử, LogP, số lượng vòng, v.v.

- **SQL (Tiếng Anh):**

```
-- Select compound id and canonical smiles
SELECT
    mol.molregno,
    mol.canonical_smiles
FROM
    molecule_dictionary mol
LIMIT 100;
```

- **Giải thích:** Đoạn SQL này truy vấn cơ sở dữ liệu ChEMBL để lấy ID hợp chất (molregno) và chuỗi SMILES tương ứng từ bảng `molecule_dictionary`.
- **Explanation:** This SQL query retrieves the compound ID (molregno) and the corresponding SMILES string from the `molecule_dictionary` table in the ChEMBL database.

- **Python:**

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

# Load data from CSV
data = pd.read_csv('your_smiles_data.csv')

# Function to calculate molecular weight
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Apply the function to the 'canonical_smiles' column
data['molecular_weight'] = data['canonical_smiles'].apply(calculate_mw)

# Print the first few rows with molecular weight
print(data.head())
```

- **Giải thích:** Đoạn Python này đọc dữ liệu SMILES từ file CSV, sử dụng RDKit để chuyển đổi chuỗi SMILES thành đối tượng phân tử, tính toán trọng lượng phân tử và lưu vào một cột mới trong DataFrame.
- **Explanation:** This Python script reads SMILES data from a CSV file, uses RDKit to convert SMILES strings into molecular objects, calculates the molecular weight, and stores it in a new column in the DataFrame.

**3. Phân tích mối tương quan giữa descriptor và hoạt tính (Descriptor-Activity Correlation Analysis):**

- **Mô tả:** Tính toán các descriptor phân tử và sau đó phân tích mối tương quan giữa các descriptor này và giá trị hoạt tính (ví dụ: IC50).

- **SQL (Tiếng Anh):**

```sql
-- Combine activity data and molecule data
SELECT
    act.standard_value,
    mol.canonical_smiles
FROM
    activities act
JOIN
    molecule_dictionary mol ON act.molregno = mol.molregno
WHERE
    act.standard_type = 'IC50'
    AND act.standard_relation = '='
LIMIT 100;
```

- **Giải thích:** Đoạn SQL này kết hợp dữ liệu hoạt tính (IC50) từ bảng `activities` với chuỗi SMILES từ bảng `molecule_dictionary` dựa trên ID hợp chất (molregno).

- **Explanation:** This SQL query combines activity data (IC50) from the `activities` table with SMILES strings from the `molecule_dictionary` table based on the compound ID (molregno).

- **Python:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np

# Load data from CSV
data = pd.read_csv('your_combined_data.csv')

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

# Remove rows with NaN in 'standard_value'
data = data.dropna(subset=['standard_value'])

# Function to calculate molecular descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol), Descriptors.MolLogP(mol)
    else:
        return None, None

# Apply the function to the 'canonical_smiles' column
data['molecular_weight'], data['logp'] =
zip(*data['canonical_smiles'].apply(calculate_descriptors))

# Remove rows where descriptor calculation failed
data = data.dropna(subset=['molecular_weight', 'logp'])

# Calculate correlation
correlation_mw_ic50 = np.corrcoef(data['molecular_weight'],
data['standard_value'])[0, 1]
correlation_logp_ic50 = np.corrcoef(data['logp'], data['standard_value'])[0, 1]

print(f"Correlation between Molecular Weight and IC50: {correlation_mw_ic50}")
print(f"Correlation between LogP and IC50: {correlation_logp_ic50}")
```

- **Giải thích:** Đoạn Python này đọc dữ liệu đã kết hợp từ file CSV, tính toán trọng lượng phân tử và LogP cho mỗi hợp chất, sau đó tính toán hệ số tương quan giữa các descriptor này và giá trị IC50.
- **Explanation:** This Python script reads the combined data from a CSV file, calculates the molecular weight and LogP for each compound, and then calculates the correlation coefficient between these descriptors and the IC50 values.

**4. Xây dựng mô hình QSAR đơn giản (Simple QSAR Model Building):**

- **Mô tả:** Xây dựng một mô hình định lượng cấu trúc-hoạt tính (QSAR) đơn giản để dự đoán hoạt tính dựa trên các descriptor phân tử.

- **Python:**

```
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

# Load data from CSV
data = pd.read_csv('your_combined_data.csv')

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

# Remove rows with NaN in 'standard_value'
data = data.dropna(subset=['standard_value'])

# Function to calculate molecular descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol), Descriptors.MolLogP(mol)
    else:
        return None, None

# Apply the function to the 'canonical_smiles' column
data['molecular_weight'], data['logp'] =
zip(*data['canonical_smiles'].apply(calculate_descriptors))

# Remove rows where descriptor calculation failed
data = data.dropna(subset=['molecular_weight', 'logp'])

# Prepare data for modeling
X = data[['molecular_weight', 'logp']]
y = data['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 and train a linear regression model
model = LinearRegression()
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) # squared=False if scikit-learn version
> 0.20
print(f"Mean Squared Error: {mse}")
```

- **Giải thích:** Đoạn Python này đọc dữ liệu, tính toán trọng lượng phân tử và LogP, chia dữ liệu thành tập huấn luyện và tập kiểm tra, xây dựng mô hình hồi quy tuyến tính và đánh giá hiệu suất của mô hình bằng Mean Squared Error (MSE).
- **Explanation:** This Python script reads the data, calculates molecular weight and LogP, splits the data into training and testing sets, builds a linear regression model, and evaluates the model's performance using Mean Squared Error (MSE).

**5. Phân tích cấu trúc con (Substructure Analysis):**

- **Mô tả:** Tìm kiếm và phân tích sự xuất hiện của các cấu trúc con cụ thể trong tập dữ liệu.

- **SQL (Tiếng Anh):** - không phù hợp, cấu trúc con phức tạp khó dùng SQL
- **Python:**

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem import AllChem

# Load data from CSV
data = pd.read_csv('your_smiles_data.csv')

# Define a substructure (e.g., benzene ring)
substructure = Chem.MolFromSmiles('c1ccccc1')

# Function to check for substructure presence
def has_substructure(smiles, substructure):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return mol.HasSubstructMatch(substructure)
    else:
        return False

# Apply the function to the 'canonical_smiles' column
data['has_benzene'] = data['canonical_smiles'].apply(lambda x:
has_substructure(x, substructure))

# Print the first few rows with substructure information
print(data.head())
```

  - **Giải thích:** Đoạn Python này đọc dữ liệu SMILES, định nghĩa một cấu trúc con (ví dụ: vòng benzen), và kiểm tra xem mỗi phân tử có chứa cấu trúc con này hay không.
  - **Explanation:** This Python script reads SMILES data, defines a substructure (e.g., benzene ring), and checks whether each molecule contains this substructure.

**Giải quyết các lỗi bạn đã đề cập:**

- **Lỗi SQL:** `ERROR: operator does not exist: numeric ~ unknown`

  Lỗi này 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 đảm bảo rằng cột `act.standard_value` thực sự là kiểu text trước khi sử dụng toán tử ~. Bạn có thể sử dụng hàm `CAST` để chuyển đổi kiểu dữ liệu:

```
AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'
```

  (Tiếng Anh: The error occurs because you're trying to use the ~ operator (regular expression match) on a numeric column. To fix this, ensure the `act.standard_value` column is of text type using `CAST` function.)

- **Lỗi Python:** `squared=False` **trong** `mean_squared_error`

  Nếu bạn đang sử dụng phiên bản scikit-learn cũ (trước 0.22), tham số `squared=False` không được hỗ trợ trong hàm `mean_squared_error`. Để khắc phục, bạn có thể tính căn bậc hai của MSE để có được RMSE (Root Mean Squared Error):

```
from sklearn.metrics import mean_squared_error
import numpy as np

mse = mean_squared_error(y_test, y_pred)
```

```
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error: {rmse}")
```

(Tiếng Anh: If you're using an older version of scikit-learn (before 0.22), the `squared=False` parameter is not supported in `mean_squared_error`. Calculate the square root of MSE to get RMSE.)

**Lưu ý quan trọng:**

- **Tiền xử lý dữ liệu:** Dữ liệu ChEMBL có thể chứa nhiều giá trị ngoại lệ và sai sót. Hãy dành thời gian để làm sạch và tiền xử lý dữ liệu một cách cẩn thận trước khi tiến hành phân tích.
- **Lựa chọn descriptor:** Lựa chọn các descriptor phân tử phù hợp là rất quan trọng để xây dựng các mô hình dự đoán chính xác. Hãy thử nghiệm với nhiều loại descriptor khác nhau và đánh giá hiệu quả của chúng.
- **Đánh giá mô hình:** Sử dụng các kỹ thuật đánh giá mô hình phù hợp (ví dụ: cross-validation) để đảm bảo rằng mô hình của bạn có khả năng khái quát hóa tốt trên dữ liệu mới.
- **Giới hạn 100 dòng:** Bạn nên bắt đầu với 100 dòng để thử nghiệm và gỡ lỗi, nhưng sau đó hãy mở rộng quy mô phân tích lên toàn bộ tập dữ liệu để có kết quả chính xác hơn.

Tôi hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án của mình một cách hiệu quả. Hãy cho tôi biết nếu bạn có bất kỳ câu hỏi nào khác!