PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Cảnh báo an toàn liên kết với hoạt tính sinh học

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit để phục vụ nghiên cứu và phát triển thuốc. Với vai trò là một chuyên gia R&D dược học, tôi sẽ giúp bạn xây dựng một quy trình làm việc hiệu quả, giải quyết các vấn đề kỹ thuật và đưa ra các ví dụ cụ thể để bạn tham khảo.

Dưới đây là phân tích, hướng dẫn chi tiết và các ví dụ code mẫu, được trình bày song ngữ (tiếng Anh và tiếng Việt) để bạn dễ dàng theo dõi:

1. Phân tích mô hình phân tích (Analysis of the Analysis Model)

Mục tiêu của bạn là khai thác dữ liệu từ ChEMBL 35, một cơ sở dữ liệu lớn chứa thông tin về các phân tử hoạt tính sinh học. Sử dụng RDKit, một bộ công cụ hóa tin học mạnh mẽ, bạn có thể tính toán các descriptor phân tử (molecular descriptors) và fingerprints để biểu diễn cấu trúc hóa học của các hợp chất. Sau đó, bạn có thể sử dụng các mô hình học máy (machine learning models) để xây dựng các mô hình QSAR/QSPR (Quantitative Structure-Activity/Property Relationship) dự đoán hoạt tính sinh học hoặc tính chất của các hợp chất mới.

Các bước chính trong quy trình phân tích:

1. **Trích xuất dữ liệu từ ChEMBL 35 (Data Extraction from ChEMBL 35):** Sử dụng SQL để truy vấn dữ liệu cần thiết từ cơ sở dữ liệu ChEMBL 35. Dữ liệu này có thể bao gồm cấu trúc phân tử (SMILES), hoạt tính sinh học (IC50, Ki, Kd, etc.), và các thông tin liên quan khác.
2. **Tiền xử lý dữ liệu (Data Preprocessing):** Làm sạch dữ liệu, xử lý các giá trị thiếu, chuẩn hóa dữ liệu hoạt tính sinh học (ví dụ: chuyển đổi tất cả về pIC50).
3. **Tính toán descriptor phân tử (Molecular Descriptor Calculation):** Sử dụng RDKit để tính toán các descriptor phân tử và fingerprints từ cấu trúc SMILES. Các descriptor này có thể bao gồm các tính chất vật lý hóa học, các thông số hình học, và các fingerprints biểu diễn cấu trúc phân tử.
4. **Lựa chọn đặc trưng (Feature Selection):** Chọn các descriptor quan trọng nhất để đưa vào mô hình học máy. Điều này có thể được thực hiện bằng các phương pháp thống kê hoặc các thuật toán lựa chọn đặc trưng.
5. **Xây dựng mô hình học máy (Machine Learning Model Building):** Sử dụng các thuật toán học máy như hồi quy tuyến tính, random forest, SVM, hoặc mạng nơ-ron để xây dựng mô hình dự đoán hoạt tính sinh học.
6. **Đánh giá mô hình (Model Evaluation):** Đánh giá hiệu suất của mô hình bằng các chỉ số phù hợp như R-squared, RMSE, MAE, AUC, etc.
7. **Giải thích mô hình (Model Interpretation):** Phân tích các descriptor quan trọng nhất ảnh hưởng đến hoạt tính sinh học để hiểu rõ hơn về mối quan hệ cấu trúc-hoạt tính.

2. Hướng dẫn song ngữ (Bilingual Guide)

2.1. Code SQL (SQL Code)

- **Mục đích (Purpose):** Trích xuất dữ liệu hoạt tính sinh học từ cơ sở dữ liệu ChEMBL 35 (Extract bioactivity data from ChEMBL 35 database).
- **Giải thích (Explanation):** Truy vấn này lấy dữ liệu từ các bảng `activities`, `activity_properties`, `assays`, `compound_structures`, và `target_dictionary` để lọc các hoạt tính dựa trên mục tiêu cụ thể và loại hoạt tính (This query retrieves data from the `activities`,

`activity_properties`, `assays`, `compound_structures`, and `target_dictionary` tables to filter activities based on a specific target and activity type).

- **Lưu ý (Note):** Sửa đổi `target_chembl_id` và các điều kiện khác để phù hợp với mục tiêu nghiên cứu của bạn (Modify `target_chembl_id` and other conditions to suit your research objectives).

```sql
-- English
-- Extract bioactivity data for a specific target, limiting to 100 rows
SELECT
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    activity_properties ap ON act.activity_id = ap.activity_id
JOIN
    assays ass ON act.assay_id = ass.assay_id
JOIN
    compound_structures cs ON act.molregno = cs.molregno
JOIN
    target_dictionary td ON ass.tid = td.tid
WHERE
    td.target_chembl_id = 'CHEMBL205'  -- Replace with your target of interest
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0  -- Avoid zero values
    AND act.standard_value < 10000 -- Limit to reasonable values
    AND act.standard_value::text ~ '^[0-9\.]+$'  -- Ensure it's a number
LIMIT 100;

-- Vietnamese
-- Trích xuất dữ liệu hoạt tính sinh học cho một mục tiêu cụ thể, giới hạn 100 dòng
SELECT
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    activity_properties ap ON act.activity_id = ap.activity_id
JOIN
    assays ass ON act.assay_id = ass.assay_id
JOIN
    compound_structures cs ON act.molregno = cs.molregno
JOIN
    target_dictionary td ON ass.tid = td.tid
WHERE
    td.target_chembl_id = 'CHEMBL205'  -- Thay thế bằng mục tiêu bạn quan tâm
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0  -- Tránh các giá trị bằng không
    AND act.standard_value < 10000 -- Giới hạn trong khoảng giá trị hợp lý
    AND act.standard_value::text ~ '^[0-9\.]+$'  -- Đảm bảo là một số
LIMIT 100;
```

Sửa lỗi SQL (Fixing SQL Error):

Lỗi `ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard_value ~ '^[0-9\.]+$'` 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 `standard_value` sang kiểu text trước khi so sánh:

```
AND act.standard_value::text ~ '^[0-9\.]+$'
```

2.2. Code Python (Python Code)

- **Mục đích (Purpose):** Đọc dữ liệu từ file CSV, tính toán descriptor phân tử bằng RDKit và hiển thị một vài dòng đầu (Read data from CSV file, calculate molecular descriptors using RDKit, and display the first few rows).
- **Giải thích (Explanation):** Đoạn code này sử dụng thư viện `pandas` để đọc dữ liệu từ file CSV. Sau đó, nó sử dụng RDKit để chuyển đổi cấu trúc SMILES thành đối tượng phân tử và tính toán descriptor `MolWt` (Molecular Weight). Cuối cùng, nó in ra 5 dòng đầu của DataFrame (This code uses the `pandas` library to read data from a CSV file. Then, it uses RDKit to convert SMILES structures into molecule objects and calculates the `MolWt` descriptor (Molecular Weight). Finally, it prints the first 5 rows of the DataFrame).
- **Lưu ý (Note):** Đảm bảo bạn đã cài đặt các thư viện `pandas` và `rdkit` trước khi chạy code (Make sure you have installed the `pandas` and `rdkit` libraries before running the code).

```
# English
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import os

# Define base path
base_path = "../data"  # Adjust if your actual base path is different

# Construct the full file path
csv_file_path = os.path.join(base_path, "your_data.csv")  # Replace with your actual
CSV file name

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

# Read the CSV file into a pandas DataFrame
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: The file {csv_file_path} was not found.")
    exit()

# Apply the function to the 'canonical_smiles' column to create a new 'mol_weight'
column
df['mol_weight'] = df['canonical_smiles'].apply(calculate_mol_weight)

# Print the first 5 rows of the DataFrame, including the new 'mol_weight' column
print(df.head())

# Vietnamese
import pandas as pd
from rdkit import Chem
```

```
from rdkit.Chem import Descriptors
import os

# Xác định đường dẫn gốc
base_path = "../data"  # Điều chỉnh nếu đường dẫn gốc thực tế của bạn khác

# Xây dựng đường dẫn đầy đủ đến file CSV
csv_file_path = os.path.join(base_path, "your_data.csv")  # Thay thế bằng tên file CSV thực tế của bạn

# Hàm tính toán khối lượng phân tử
def calculate_mol_weight(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Đọc file CSV vào một DataFrame của pandas
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file {csv_file_path}.")
    exit()

# Áp dụng hàm cho cột 'canonical_smiles' để tạo cột mới 'mol_weight'
df['mol_weight'] = df['canonical_smiles'].apply(calculate_mol_weight)

# In 5 dòng đầu tiên của DataFrame, bao gồm cột 'mol_weight' mới
print(df.head())
```

Sửa lỗi Python (Fixing Python Error):

Thông báo lỗi "phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean_squared_error" cho biết rằng phiên bản scikit-learn bạn đang sử dụng quá cũ và không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`. Để khắc phục, bạn có hai lựa chọn:

1. **Nâng cấp scikit-learn (Upgrade scikit-learn):** Đây là cách tốt nhất. Sử dụng pip để nâng cấp lên phiên bản mới nhất:

   ```
   pip install --upgrade scikit-learn
   ```

2. **Tính RMSE thủ công (Calculate RMSE manually):** Nếu bạn không thể nâng cấp scikit-learn, bạn có thể tính căn bậc hai của MSE để có được RMSE:

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

   # Tính MSE
   mse = mean_squared_error(y_true, y_pred)

   # Tính RMSE bằng cách lấy căn bậc hai của MSE
   rmse = np.sqrt(mse)
   print(f"RMSE: {rmse}")
   ```

3. Ví dụ code (Code Examples)

Dưới đây là 5 ví dụ code SQL và Python mẫu, minh họa các tác vụ khác nhau trong quy trình phân tích dữ liệu ChEMBL 35:

Ví dụ 1: Trích xuất dữ liệu và tính logP (Example 1: Data Extraction and logP Calculation)

- **SQL (English/Vietnamese):**

```sql
-- English
-- Extract SMILES and IC50 values for a specific target and calculate logP using RDKit
in Python
SELECT
    cs.canonical_smiles,
    act.standard_value
FROM
    activities act
JOIN
    activity_properties ap ON act.activity_id = ap.activity_id
JOIN
    assays ass ON act.assay_id = ass.assay_id
JOIN
    compound_structures cs ON act.molregno = cs.molregno
JOIN
    target_dictionary td ON ass.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
LIMIT 100;

-- Vietnamese
-- Trích xuất giá trị SMILES và IC50 cho một mục tiêu cụ thể và tính logP bằng RDKit
trong Python
SELECT
    cs.canonical_smiles,
    act.standard_value
FROM
    activities act
JOIN
    activity_properties ap ON act.activity_id = ap.activity_id
JOIN
    assays ass ON act.assay_id = ass.assay_id
JOIN
    compound_structures cs ON act.molregno = cs.molregno
JOIN
    target_dictionary td ON ass.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
LIMIT 100;
```

- **Python (English/Vietnamese):**

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

# Define base path
base_path = "../data"  # Adjust if your actual base path is different

# Construct the full file path
csv_file_path = os.path.join(base_path, "your_data.csv")  # Replace with your actual
CSV file name
```

```python
# Function to calculate logP
def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolLogP(mol)
    else:
        return None

# Read the CSV file into a pandas DataFrame
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: The file {csv_file_path} was not found.")
    exit()

# Apply the function to the 'canonical_smiles' column to create a new 'logp' column
df['logp'] = df['canonical_smiles'].apply(calculate_logp)

# Print the first 5 rows of the DataFrame, including the new 'logp' column
print(df.head())

# Vietnamese
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import os

# Xác định đường dẫn gốc
base_path = "../data"  # Điều chỉnh nếu đường dẫn gốc thực tế của bạn khác

# Xây dựng đường dẫn đầy đủ đến file CSV
csv_file_path = os.path.join(base_path, "your_data.csv")  # Thay thế bằng tên file CSV thực tế của bạn

# Hàm tính toán logP
def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolLogP(mol)
    else:
        return None

# Đọc file CSV vào một DataFrame của pandas
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file {csv_file_path}.")
    exit()

# Áp dụng hàm cho cột 'canonical_smiles' để tạo cột mới 'logp'
df['logp'] = df['canonical_smiles'].apply(calculate_logp)

# In 5 dòng đầu tiên của DataFrame, bao gồm cột 'logp' mới
print(df.head())
```

Ví dụ 2: Tính toán fingerprints (Example 2: Fingerprint Calculation)

- **Python (English/Vietnamese):**

```python
# English
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
import os

# Define base path
base_path = "../data"  # Adjust if your actual base path is different

# Construct the full file path
csv_file_path = os.path.join(base_path, "your_data.csv")  # Replace with your actual
CSV file name

# Function to calculate Morgan fingerprints
def calculate_morgan_fingerprint(smiles, radius=2, nBits=2048):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits)
    else:
        return None

# Read the CSV file into a pandas DataFrame
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: The file {csv_file_path} was not found.")
    exit()

# Apply the function to the 'canonical_smiles' column to create a new 'morgan_fp'
column
df['morgan_fp'] = df['canonical_smiles'].apply(calculate_morgan_fingerprint)

# Print the first 5 rows of the DataFrame, including the new 'morgan_fp' column
print(df.head())

# Vietnamese
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
import os

# Xác định đường dẫn gốc
base_path = "../data"  # Điều chỉnh nếu đường dẫn gốc thực tế của bạn khác

# Xây dựng đường dẫn đầy đủ đến file CSV
csv_file_path = os.path.join(base_path, "your_data.csv")  # Thay thế bằng tên file CSV
thực tế của bạn

# Hàm tính toán Morgan fingerprints
def calculate_morgan_fingerprint(smiles, radius=2, nBits=2048):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits)
    else:
        return None

# Đọc file CSV vào một DataFrame của pandas
try:
    df = pd.read_csv(csv_file_path)
```

```
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file {csv_file_path}.")
    exit()

# Áp dụng hàm cho cột 'canonical_smiles' để tạo cột mới 'morgan_fp'
df['morgan_fp'] = df['canonical_smiles'].apply(calculate_morgan_fingerprint)

# In 5 dòng đầu tiên của DataFrame, bao gồm cột 'morgan_fp' mới
print(df.head())
```

Ví dụ 3: Chuẩn hóa dữ liệu hoạt tính sinh học (Example 3: Bioactivity Data Normalization)

- **Python (English/Vietnamese):**

```
# English
import pandas as pd
import numpy as np
import os

# Define base path
base_path = "../data"  # Adjust if your actual base path is different

# Construct the full file path
csv_file_path = os.path.join(base_path, "your_data.csv")  # Replace with your actual
CSV file name

# Function to convert IC50 to pIC50
def ic50_to_pic50(ic50):
    if ic50:
        try:
            ic50 = float(ic50)
            if ic50 > 0:
                pIC50 = -np.log10(ic50 * 1e-9)  # Convert nM to M and then to pIC50
                return pIC50
            else:
                return None
        except ValueError:
            return None
    else:
        return None

# Read the CSV file into a pandas DataFrame
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: The file {csv_file_path} was not found.")
    exit()

# Apply the function to the 'standard_value' column to create a new 'pIC50' column
df['pIC50'] = df['standard_value'].apply(ic50_to_pic50)

# Print the first 5 rows of the DataFrame, including the new 'pIC50' column
print(df.head())

# Vietnamese
import pandas as pd
import numpy as np
import os

# Xác định đường dẫn gốc
base_path = "../data"  # Điều chỉnh nếu đường dẫn gốc thực tế của bạn khác
```

```
# Xây dựng đường dẫn đầy đủ đến file CSV
csv_file_path = os.path.join(base_path, "your_data.csv")  # Thay thế bằng tên file CSV
thực tế của bạn

# Hàm chuyển đổi IC50 sang pIC50
def ic50_to_pic50(ic50):
    if ic50:
        try:
            ic50 = float(ic50)
            if ic50 > 0:
                pIC50 = -np.log10(ic50 * 1e-9)  # Chuyển đổi nM sang M và sau đó sang
pIC50
                return pIC50
            else:
                return None
        except ValueError:
            return None
    else:
        return None

# Đọc file CSV vào một DataFrame của pandas
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file {csv_file_path}.")
    exit()

# Áp dụng hàm cho cột 'standard_value' để tạo cột mới 'pIC50'
df['pIC50'] = df['standard_value'].apply(ic50_to_pic50)

# In 5 dòng đầu tiên của DataFrame, bao gồm cột 'pIC50' mới
print(df.head())
```

Ví dụ 4: Xây dựng mô hình hồi quy tuyến tính (Example 4: Linear Regression Model Building)

- **Python (English/Vietnamese):**

```
# English
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
import os

# Define base path
base_path = "../data"  # Adjust if your actual base path is different

# Construct the full file path
csv_file_path = os.path.join(base_path, "your_data.csv")  # Replace with your actual
CSV file name

# Function to calculate molecular weight
def calculate_mol_weight(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
```

```
        return None

# Read the CSV file into a pandas DataFrame
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: The file {csv_file_path} was not found.")
    exit()

# Calculate molecular weight
df['mol_weight'] = df['canonical_smiles'].apply(calculate_mol_weight)

# Data preprocessing: Drop rows with missing values
df = df.dropna(subset=['mol_weight', 'standard_value'])

# Prepare data for the model
X = df[['mol_weight']]  # Features
y = df['standard_value']  # Target variable

# 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 the 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)
rmse = np.sqrt(mse)  # Calculate RMSE manually
print(f"Root Mean Squared Error: {rmse}")


# Vietnamese
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
import os

# Xác định đường dẫn gốc
base_path = "../data"  # Điều chỉnh nếu đường dẫn gốc thực tế của bạn khác

# Xây dựng đường dẫn đầy đủ đến file CSV
csv_file_path = os.path.join(base_path, "your_data.csv")  # Thay thế bằng tên file CSV
thực tế của bạn

# Hàm tính toán khối lượng phân tử
def calculate_mol_weight(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    else:
```

```
        return None

# Đọc file CSV vào một DataFrame của pandas
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file {csv_file_path}.")
    exit()

# Tính toán khối lượng phân tử
df['mol_weight'] = df['canonical_smiles'].apply(calculate_mol_weight)

# Tiền xử lý dữ liệu: Loại bỏ các hàng có giá trị bị thiếu
df = df.dropna(subset=['mol_weight', 'standard_value'])

# Chuẩn bị dữ liệu cho mô hình
X = df[['mol_weight']]  # Các đặc trưng
y = df['standard_value']  # Biến mục tiêu

# 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)

# Tạo và huấn luyện mô hình hồi quy tuyến tính
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)
rmse = np.sqrt(mse)  # Tính RMSE thủ công
print(f"Sai số bình phương gốc trung bình: {rmse}")
```

Ví dụ 5: Sử dụng Random Forest để dự đoán hoạt tính (Example 5: Using Random Forest for Activity Prediction)

- **Python (English/Vietnamese):**

```
# English
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
import os

# Define base path
base_path = "../data"  # Adjust if your actual base path is different

# Construct the full file path
csv_file_path = os.path.join(base_path, "your_data.csv")  # Replace with your actual
CSV file name

# Function to calculate molecular weight
def calculate_mol_weight(smiles):
    mol = Chem.MolFromSmiles(smiles)
```

```
    if mol:
        return Descriptors.MolWt(mol)
    else:
        return None

# Read the CSV file into a pandas DataFrame
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: The file {csv_file_path} was not found.")
    exit()

# Calculate molecular weight
df['mol_weight'] = df['canonical_smiles'].apply(calculate_mol_weight)

# Data preprocessing: Drop rows with missing values
df = df.dropna(subset=['mol_weight', 'standard_value'])

# Prepare data for the model
X = df[['mol_weight']]  # Features
y = df['standard_value']  # Target variable

# 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 the Random Forest Regressor model
model = RandomForestRegressor(n_estimators=100, random_state=42)
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)
rmse = np.sqrt(mse)  # Calculate RMSE manually
print(f"Root Mean Squared Error: {rmse}")

# Vietnamese
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
import os

# Xác định đường dẫn gốc
base_path = "../data"  # Điều chỉnh nếu đường dẫn gốc thực tế của bạn khác

# Xây dựng đường dẫn đầy đủ đến file CSV
csv_file_path = os.path.join(base_path, "your_data.csv")  # Thay thế bằng tên file CSV
thực tế của bạn

# Hàm tính toán khối lượng phân tử
def calculate_mol_weight(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
```

```
        return Descriptors.MolWt(mol)
    else:
        return None

# Đọc file CSV vào một DataFrame của pandas
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file {csv_file_path}.")
    exit()

# Tính toán khối lượng phân tử
df['mol_weight'] = df['canonical_smiles'].apply(calculate_mol_weight)

# Tiền xử lý dữ liệu: Loại bỏ các hàng có giá trị bị thiếu
df = df.dropna(subset=['mol_weight', 'standard_value'])

# Chuẩn bị dữ liệu cho mô hình
X = df[['mol_weight']]  # Các đặc trưng
y = df['standard_value']  # Biến mục tiêu

# 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)

# Tạo và huấn luyện mô hình Random Forest Regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
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)
rmse = np.sqrt(mse)  # Tính RMSE thủ công
print(f"Sai số bình phương gốc trung bình: {rmse}")
```

Lưu ý quan trọng:

- **Đường dẫn file:** Hãy chắc chắn rằng bạn đã thay đổi đường dẫn file CSV (`csv_file_path`) trong các ví dụ Python để trỏ đến vị trí chính xác của file dữ liệu của bạn.
- **Chọn descriptor:** Trong các ví dụ trên, tôi chỉ sử dụng `MolWt` (khối lượng phân tử) làm một descriptor đơn giản. Trong thực tế, bạn nên tính toán và sử dụng nhiều descriptor khác nhau để mô hình có thể học được các mối quan hệ phức tạp hơn giữa cấu trúc và hoạt tính.
- **Tối ưu hóa mô hình:** Các mô hình học máy trong các ví dụ trên chỉ là các ví dụ cơ bản. Bạn nên thử nghiệm với các thuật toán khác nhau, điều chỉnh các siêu tham số và sử dụng các kỹ thuật như cross-validation để tối ưu hóa hiệu suất của mô hình.
- **Đảm bảo kết nối:** Đảm bảo rằng máy chủ PostgreSQL của bạn đang chạy và có thể truy cập được từ máy tính mà bạn đang chạy Jupyter Notebook.
- **AIMLOps:** Nhớ tuân thủ cấu trúc thư mục AIMLOps Template để dự án của bạn được tổ chức một cách chuyên nghiệp.

Tôi 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 khác, đừng ngần ngại hỏi! Chúc bạn thành công!