PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: QSAR with Random Forest and ChEMBL descriptors

Okay, I understand. You're working with ChEMBL 35 data, using RDKit for analysis, and following an AIMLops template. You have a PostgreSQL database, and you'll be using Jupyter notebooks for your analysis. You've also encountered a couple of errors and want to limit the data to 100 rows for quicker processing. I will provide analysis, guidance, SQL and Python code examples.

Here's a breakdown of the approach, code, and explanations in both English and Vietnamese:

1. Analysis and Model Explanation

English:

The project aims to analyze ChEMBL 35 data for drug discovery. The initial steps involve querying the PostgreSQL database to extract relevant information, such as molecule structures (SMILES) and activity data (IC50, Ki, etc.). This extracted data is then loaded into a Pandas DataFrame within a Jupyter notebook. We will then utilize RDKit to generate molecular descriptors from the SMILES strings.

Possible analytical models include:

- **Exploratory Data Analysis (EDA):** Visualize activity distributions, identify potential outliers, and explore relationships between different activity types.
- **Structure-Activity Relationship (SAR) modeling:** Use machine learning models (e.g., Random Forest, Support Vector Machines) to predict activity based on molecular descriptors.
- **Clustering:** Group molecules based on structural similarity or activity profiles.
- **Filtering and Prioritization:** Identify promising compounds based on activity thresholds and desired properties.

The example code focuses on extracting basic data and generating some simple molecular descriptors. More complex modeling would require further feature engineering and model selection.

Vietnamese:

Dự án này nhằm mục đích phân tích dữ liệu ChEMBL 35 cho việc khám phá thuốc. Các bước ban đầu bao gồm truy vấn cơ sở dữ liệu PostgreSQL để trích xuất thông tin liên quan, chẳng hạn như cấu trúc phân tử (SMILES) và dữ liệu hoạt động (IC50, Ki, v.v.). Dữ liệu được trích xuất này sau đó được tải vào DataFrame Pandas trong một Jupyter notebook. Sau đó, chúng ta sẽ sử dụng RDKit để tạo ra các mô tả phân tử từ chuỗi SMILES.

Các mô hình phân tích có thể bao gồm:

- **Phân tích dữ liệu thăm dò (EDA):** Trực quan hóa phân phối hoạt động, xác định các giá trị ngoại lai tiềm năng và khám phá mối quan hệ giữa các loại hoạt động khác nhau.
- **Mô hình hóa mối quan hệ cấu trúc-hoạt động (SAR):** Sử dụng các mô hình học máy (ví dụ: Random Forest, Support Vector Machines) để dự đoán hoạt động dựa trên các mô tả phân tử.
- **Phân cụm:** Nhóm các phân tử dựa trên sự tương đồng về cấu trúc hoặc hồ sơ hoạt động.
- **Lọc và ưu tiên:** Xác định các hợp chất đầy hứa hẹn dựa trên ngưỡng hoạt động và các thuộc tính mong muốn.

Đoạn mã ví dụ tập trung vào việc trích xuất dữ liệu cơ bản và tạo ra một số mô tả phân tử đơn giản. Mô hình hóa phức tạp hơn sẽ yêu cầu kỹ thuật đặc trưng và lựa chọn mô hình hơn nữa.

2. SQL Code (Creating CSV File)

English:

This SQL code retrieves activity data and corresponding SMILES strings for a specified target. It also addresses the error you encountered regarding the regular expression by casting the `standard_value` column to `TEXT` before applying the regex. Critically, it also limits the results to 100 rows. Remember to run this code in pgAdmin and save the output as a CSV file in your `/data` directory.

```
-- Save this as /data/chembl_35_activity_data.csv
COPY (
    SELECT
        act.molregno,
        act.standard_type,
        act.standard_value,
        act.standard_units,
        md.canonical_smiles
    FROM
        activities act
    JOIN
        molecule_dictionary md ON act.molregno = md.molregno
    WHERE
        act.standard_type = 'IC50'  -- Example: You can change this to Ki, EC50, etc.
        AND act.standard_relation = '='
        AND act.standard_value IS NOT NULL
        AND act.standard_units = 'nM'
        AND CAST(act.standard_value AS TEXT) ~ '^[0-9.]+$' -- Corrected regex
        AND act.target_id IN (SELECT target_id FROM target_dictionary WHERE pref_name
= 'Epidermal Growth Factor Receptor') -- Example Target
    LIMIT 100
)
TO '/tmp/chembl_35_activity_data.csv' WITH CSV HEADER;
```

Important Notes for SQL Code:

- **`standard_type`:** Modify the `WHERE act.standard_type = 'IC50'` clause to filter for your desired activity type (e.g., `'Ki'`, `'EC50'`).
- **`target_id`**: Modify the `target_id` clause to select the data related to the target you want to analyze. The example uses `'Epidermal Growth Factor Receptor'`. You can find appropriate `pref_name` values in the `target_dictionary` table.
- **File Path:** Change `/tmp/chembl_35_activity_data.csv` to the correct path within your AIMLops folder structure. For example, it might be `/app/data/chembl_35_activity_data.csv`. **IMPORTANT:** The PostgreSQL server user (usually `postgres`) needs write permissions to this directory.
- **Regex:** The `CAST(act.standard_value AS TEXT) ~ '^[0-9.]+$'` part is crucial. It first casts the `standard_value` to a text data type before applying the regular expression to check if it contains only numbers and periods.
- **`LIMIT 100`:** This limits the result set to 100 rows. Remove or adjust this as needed.

Vietnamese:

Đoạn mã SQL này truy xuất dữ liệu hoạt động và chuỗi SMILES tương ứng cho một mục tiêu cụ thể. Nó cũng giải quyết lỗi mà bạn gặp phải liên quan đến biểu thức chính quy bằng cách chuyển đổi cột `standard_value` thành `TEXT` trước khi áp dụng biểu thức chính quy. Quan trọng nhất, nó cũng giới hạn kết quả thành 100 hàng. Hãy nhớ chạy mã này trong pgAdmin và lưu đầu ra dưới dạng tệp CSV trong thư mục `/data` của bạn.

```
-- Lưu cái này thành /data/chembl_35_activity_data.csv
COPY (
```

```
    SELECT
        act.molregno,
        act.standard_type,
        act.standard_value,
        act.standard_units,
        md.canonical_smiles
    FROM
        activities act
    JOIN
        molecule_dictionary md ON act.molregno = md.molregno
    WHERE
        act.standard_type = 'IC50'  -- Ví dụ: Bạn có thể thay đổi cái này thành Ki,
EC50, v.v.
        AND act.standard_relation = '='
        AND act.standard_value IS NOT NULL
        AND act.standard_units = 'nM'
        AND CAST(act.standard_value AS TEXT) ~ '^[0-9.]+$' -- Regex đã sửa
        AND act.target_id IN (SELECT target_id FROM target_dictionary WHERE pref_name
= 'Epidermal Growth Factor Receptor') -- Ví dụ về mục tiêu
    LIMIT 100
)
TO '/tmp/chembl_35_activity_data.csv' WITH CSV HEADER;
```

Lưu ý quan trọng cho mã SQL:

- **`standard_type`:** Sửa đổi mệnh đề `WHERE act.standard_type = 'IC50'` để lọc theo loại hoạt động mong muốn của bạn (ví dụ: `'Ki'`, `'EC50'`).
- **`target_id`**: Sửa đổi mệnh đề `target_id` để chọn dữ liệu liên quan đến mục tiêu bạn muốn phân tích. Ví dụ sử dụng `'Epidermal Growth Factor Receptor'`. Bạn có thể tìm thấy các giá trị `pref_name` phù hợp trong bảng `target_dictionary`.
- **Đường dẫn tệp:** Thay đổi `/tmp/chembl_35_activity_data.csv` thành đường dẫn chính xác trong cấu trúc thư mục AIMLops của bạn. Ví dụ: nó có thể là `/app/data/chembl_35_activity_data.csv`. **QUAN TRỌNG:** Người dùng máy chủ PostgreSQL (thường là `postgres`) cần có quyền ghi vào thư mục này.
- **Regex:** Phần `CAST(act.standard_value AS TEXT) ~ '^[0-9.]+$'` là rất quan trọng. Nó đầu tiên chuyển đổi `standard_value` thành kiểu dữ liệu văn bản trước khi áp dụng biểu thức chính quy để kiểm tra xem nó chỉ chứa số và dấu chấm hay không.
- **`LIMIT 100`:** Điều này giới hạn tập kết quả thành 100 hàng. Xóa hoặc điều chỉnh cái này nếu cần.

3. Python Code (Jupyter Notebook: `Topic_CheMBL_35_61_1_data_loading_and_descriptor_generation.ipynb`)

English:

This Python code reads the CSV file created by the SQL query, uses RDKit to generate some basic molecular descriptors, and performs a simple data visualization. It also includes a workaround for the `squared=False` error in older scikit-learn versions.

```
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem import Lipinski
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
import numpy as np

# Define base path based on your AIMLops structure
base_path = "/app" # Adjust this to your actual base path
```

```python
# Construct the data file path
data_file = os.path.join(base_path, "data", "chembl_35_activity_data.csv")

# Load the data
try:
    df = pd.read_csv(data_file)
except FileNotFoundError:
    print(f"Error: File not found at {data_file}.  Make sure you ran the SQL query and
saved the CSV to the correct location.")
    exit()

print(f"Loaded {len(df)} rows of data.")
print(df.head())

# Create RDKit Mol objects
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))
df = df.dropna(subset=['mol']) # Remove rows where mol is None (invalid SMILES)

# Calculate some descriptors
def calculate_descriptors(mol):
    try:
        mw = Descriptors.MolWt(mol)
        logp = Descriptors.MolLogP(mol)
        hbd = Lipinski.NumHDonors(mol)
        hba = Lipinski.NumHAcceptors(mol)
        return pd.Series([mw, logp, hbd, hba])
    except:
        return pd.Series([None, None, None, None]) #Handle potential errors in
descriptor calculation

df[['mol_wt', 'logp', 'hbd', 'hba']] = df['mol'].apply(calculate_descriptors)

# Convert standard_value to numeric and handle errors
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df = df.dropna(subset=['standard_value']) # Remove rows where standard_value is NaN

# Data Visualization (Example)
plt.figure(figsize=(8, 6))
plt.hist(df['standard_value'], bins=50)
plt.xlabel('IC50 (nM)')
plt.ylabel('Frequency')
plt.title('Distribution of IC50 Values')
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(df['mol_wt'], df['logp'])
plt.xlabel('Molecular Weight')
plt.ylabel('LogP')
plt.title('Molecular Weight vs. LogP')
plt.show()

# Example: Simple Linear Regression (Demonstration - Requires further feature
engineering)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Prepare data for the model
X = df[['mol_wt', 'logp', 'hbd', 'hba']].fillna(0) # Handle missing values
```

```
(important!)
y = df['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)

# Train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model (Handle the squared=False issue)
try:
    mse = mean_squared_error(y_test, y_pred, squared=False) # Python 3.9+
except TypeError:
    mse = mean_squared_error(y_test, y_pred) # Older scikit-learn
    print("Warning:  Using older scikit-learn version.  Consider upgrading for correct
MSE calculation.")

print(f"Mean Squared Error: {mse}")
```

Explanation of Python Code:

1. **Import Libraries:** Imports necessary libraries like `os`, `pandas`, `rdkit`, `matplotlib`, and `scikit-learn`.
2. **Define Base Path:** Sets the base path for your project, ensuring the code can find the data file. **Important:** Modify this to your actual base path.
3. **Load Data:** Reads the CSV file into a Pandas DataFrame. Includes error handling if the file is not found.
4. **Create RDKit Mol Objects:** Converts SMILES strings to RDKit `Mol` objects, which are used for descriptor calculation. Rows with invalid SMILES are removed.
5. **Calculate Descriptors:** Calculates molecular weight, LogP, number of hydrogen bond donors, and number of hydrogen bond acceptors using RDKit functions.
6. **Data Cleaning:** Converts the `standard_value` column to numeric, handling potential errors and removing rows with missing values.
7. **Data Visualization:** Creates a histogram of IC50 values and a scatter plot of molecular weight vs. LogP.
8. **Simple Linear Regression (Example):**
    - Prepares the data for a linear regression model.
    - Splits the data into training and testing sets.
    - Trains a linear regression model.
    - Makes predictions.
    - Calculates the Mean Squared Error (MSE). Includes a workaround for the `squared=False` parameter in older scikit-learn versions. If you're using an older version, it will print a warning message.

Vietnamese:

Đoạn mã Python này đọc tệp CSV được tạo bởi truy vấn SQL, sử dụng RDKit để tạo ra một số mô tả phân tử cơ bản và thực hiện trực quan hóa dữ liệu đơn giản. Nó cũng bao gồm một giải pháp cho lỗi `squared=False` trong các phiên bản scikit-learn cũ hơn.

```
import os
import pandas as pd
```

```
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem import Lipinski
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
import numpy as np

# Xác định đường dẫn gốc dựa trên cấu trúc AIMLops của bạn
base_path = "/app" # Điều chỉnh cái này theo đường dẫn gốc thực tế của bạn

# Xây dựng đường dẫn tệp dữ liệu
data_file = os.path.join(base_path, "data", "chembl_35_activity_data.csv")

# Tải dữ liệu
try:
    df = pd.read_csv(data_file)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy tệp tại {data_file}. Đảm bảo bạn đã chạy truy vấn SQL
và lưu CSV vào đúng vị trí.")
    exit()

print(f"Đã tải {len(df)} hàng dữ liệu.")
print(df.head())

# Tạo đối tượng Mol RDKit
df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))
df = df.dropna(subset=['mol']) # Xóa các hàng mà mol là None (SMILES không hợp lệ)

# Tính toán một số mô tả
def calculate_descriptors(mol):
    try:
        mw = Descriptors.MolWt(mol)
        logp = Descriptors.MolLogP(mol)
        hbd = Lipinski.NumHDonors(mol)
        hba = Lipinski.NumHAcceptors(mol)
        return pd.Series([mw, logp, hbd, hba])
    except:
        return pd.Series([None, None, None, None]) # Xử lý các lỗi tiềm ẩn trong tính
toán mô tả

df[['mol_wt', 'logp', 'hbd', 'hba']] = df['mol'].apply(calculate_descriptors)

# Chuyển đổi standard_value thành số và xử lý lỗi
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df = df.dropna(subset=['standard_value']) # Xóa các hàng mà standard_value là NaN

# Trực quan hóa dữ liệu (Ví dụ)
plt.figure(figsize=(8, 6))
plt.hist(df['standard_value'], bins=50)
plt.xlabel('IC50 (nM)')
plt.ylabel('Tần số')
plt.title('Phân phối các giá trị IC50')
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(df['mol_wt'], df['logp'])
plt.xlabel('Khối lượng phân tử')
plt.ylabel('LogP')
plt.title('Khối lượng phân tử so với LogP')
```

```
plt.show()

# Ví dụ: Hồi quy tuyến tính đơn giản (Minh họa - Yêu cầu kỹ thuật đặc trưng hơn nữa)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Chuẩn bị dữ liệu cho mô hình
X = df[['mol_wt', 'logp', 'hbd', 'hba']].fillna(0) # Xử lý các giá trị bị thiếu (quan
trọng!)
y = df['standard_value']

# 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)

# Huấn luyện mô hình
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán
y_pred = model.predict(X_test)

# Đánh giá mô hình (Xử lý vấn đề squared=False)
try:
    mse = mean_squared_error(y_test, y_pred, squared=False) # Python 3.9+
except TypeError:
    mse = mean_squared_error(y_test, y_pred) # Scikit-learn cũ hơn
    print("Cảnh báo: Đang sử dụng phiên bản scikit-learn cũ hơn. Cân nhắc nâng cấp để
tính toán MSE chính xác.")

print(f"Lỗi bình phương trung bình: {mse}")
```

Giải thích mã Python:

1. **Nhập thư viện:** Nhập các thư viện cần thiết như `os`, `pandas`, `rdkit`, `matplotlib` và `scikit-learn`.
2. **Xác định đường dẫn gốc:** Đặt đường dẫn gốc cho dự án của bạn, đảm bảo mã có thể tìm thấy tệp dữ liệu. **Quan trọng:** Sửa đổi cái này theo đường dẫn gốc thực tế của bạn.
3. **Tải dữ liệu:** Đọc tệp CSV vào DataFrame Pandas. Bao gồm xử lý lỗi nếu không tìm thấy tệp.
4. **Tạo đối tượng Mol RDKit:** Chuyển đổi chuỗi SMILES thành đối tượng `Mol` RDKit, được sử dụng để tính toán mô tả. Các hàng có SMILES không hợp lệ sẽ bị xóa.
5. **Tính toán mô tả:** Tính toán trọng lượng phân tử, LogP, số lượng người cho liên kết hydro và số lượng người nhận liên kết hydro bằng các hàm RDKit.
6. **Làm sạch dữ liệu:** Chuyển đổi cột `standard_value` thành số, xử lý các lỗi tiềm ẩn và xóa các hàng có giá trị bị thiếu.
7. **Trực quan hóa dữ liệu:** Tạo biểu đồ tần suất của các giá trị IC50 và biểu đồ phân tán của trọng lượng phân tử so với LogP.
8. **Hồi quy tuyến tính đơn giản (Ví dụ):**
    - Chuẩn bị dữ liệu cho mô hình hồi quy tuyến tính.
    - Chia dữ liệu thành tập huấn luyện và tập kiểm tra.
    - Huấn luyện một mô hình hồi quy tuyến tính.
    - Đưa ra dự đoán.
    - Tính toán Lỗi bình phương trung bình (MSE). Bao gồm một giải pháp cho tham số `squared=False` trong các phiên bản scikit-learn cũ hơn. Nếu bạn đang sử dụng phiên bản cũ hơn, nó sẽ in ra một thông báo cảnh báo.

4. Running the Code

1. **SQL:** Execute the SQL query in pgAdmin and save the result as a CSV file in the `/data` folder of your AIMLops project structure (e.g., `/app/data/chembl_35_activity_data.csv`). Remember to adjust the SQL query to target the specific activity type and target of interest. **Double-check the file path!**

2. **Python:** Open the Jupyter notebook (`Topic_CheMBL_35_61_1_data_loading_and_descriptor_generation.ipynb`) and run the cells sequentially. Make sure to adjust the `base_path` variable to reflect your project's directory structure.

3. **Dependencies:** Ensure you have the necessary Python libraries installed: `pandas`, `rdkit`, `matplotlib`, and `scikit-learn`. You can install them using `pip`:

   ```
   pip install pandas rdkit matplotlib scikit-learn
   ```

5. Five Examples (Expanding on the Core Code)

Here are five example scenarios, with snippets of code that you can add to your Jupyter Notebook. These examples build upon the base code provided above. **Remember to execute the core code (data loading, descriptor generation) *before* running these examples.**

Example 1: Filtering by Molecular Weight

This example filters the DataFrame to only include molecules with a molecular weight between 200 and 500.

```
# Filter by molecular weight
df_filtered_mw = df[(df['mol_wt'] >= 200) & (df['mol_wt'] <= 500)]
print(f"Number of molecules after filtering by molecular weight:
{len(df_filtered_mw)}")
print(df_filtered_mw.head())
```

Example 2: Analyzing Activity Distribution for a Specific Target

(Requires modifying the SQL query to select a specific target and activity type) This shows the distribution of activity (e.g. IC50) for that one target.

```
import seaborn as sns

#Plot distribution of IC50 values
plt.figure(figsize=(10,6))
sns.histplot(df['standard_value'], kde=True)
plt.title("IC50 Distribution for Selected Target")
plt.xlabel("IC50 (nM)")
plt.ylabel("Frequency")
plt.show()
```

Example 3: Calculating and Visualizing TPSA (Topological Polar Surface Area)

This calculates TPSA, a descriptor related to drug permeability, and visualizes it against LogP.

```
from rdkit.Chem import rdMolDescriptors

def calculate_tpsa(mol):
    try:
        tpsa = rdMolDescriptors.CalcTPSA(mol)
        return tpsa
    except:
        return None
```

```
df['tpsa'] = df['mol'].apply(calculate_tpsa)
df = df.dropna(subset=['tpsa'])

plt.figure(figsize=(8, 6))
plt.scatter(df['logp'], df['tpsa'])
plt.xlabel('LogP')
plt.ylabel('TPSA')
plt.title('LogP vs. TPSA')
plt.show()
```

Example 4: Calculating Similarity to a Known Active Compound

This calculates the Tanimoto similarity between each molecule and a reference compound (using Morgan fingerprints).

```
from rdkit import DataStructs
from rdkit.Chem import AllChem

# Define a reference SMILES (e.g., a known active compound)
reference_smiles = "CC(=O)Oc1ccccc1C(=O)O"  # Example: Aspirin
reference_mol = Chem.MolFromSmiles(reference_smiles)
reference_fp = AllChem.GetMorganFingerprintAsBitVect(reference_mol, 2, nBits=2048)

def calculate_similarity(mol):
    try:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
        similarity = DataStructs.TanimotoSimilarity(reference_fp, fp)
        return similarity
    except:
        return None

df['similarity'] = df['mol'].apply(calculate_similarity)
df = df.dropna(subset=['similarity'])

plt.figure(figsize=(8, 6))
plt.hist(df['similarity'], bins=50)
plt.xlabel('Tanimoto Similarity to Aspirin')
plt.ylabel('Frequency')
plt.title('Distribution of Similarity Scores')
plt.show()
```

Example 5: Applying a Lipinski's Rule of Five Filter

This filters the DataFrame based on Lipinski's Rule of Five, a set of guidelines for drug-likeness.

```
# Apply Lipinski's Rule of Five
df_lipinski = df[
    (df['mol_wt'] <= 500) &
    (df['logp'] <= 5) &
    (df['hbd'] <= 5) &
    (df['hba'] <= 10)
]

print(f"Number of molecules after applying Lipinski's Rule of Five:
{len(df_lipinski)}")
print(df_lipinski.head())
```

Remember to adapt the code to your specific research question, target, and data. Good luck! Let me know if you have more questions.