PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Disease association of ChEMBL targets

Okay, I understand. You're working on a Chembl 35 data analysis project using RDKit and need help structuring your code, handling errors, and generating meaningful results within the AIMLops framework. I'll help you analyze the issues, provide corrected code, and suggest analysis techniques.

Here's a breakdown in English and Vietnamese:

Analysis Model (Phân tích Mô hình):

The core of your project lies in analyzing the ChEMBL 35 dataset. You are aiming to use SQL to retrieve relevant data (likely activity data) and then use Python (with RDKit and potentially scikit-learn) to process and analyze that data. This typically involves:

1. **Data Retrieval (Truy xuất dữ liệu):** Using SQL to query the ChEMBL database and extract relevant activity data, compound properties, and target information. This is your starting point. Focus on specific targets or activity types to keep the initial data volume manageable.
2. **Data Cleaning & Preparation (Làm sạch và Chuẩn bị dữ liệu):** The data from ChEMBL can be noisy. You'll need to:
    - Handle missing values.
    - Convert data types (e.g., standard_value to numeric).
    - Filter out irrelevant activity types.
    - Address the error related to the `numeric ~ unknown` operator. This means the `standard_value` column, despite being activity values, might be stored as text/strings.
3. **Feature Generation (Tạo Đặc trưng):** Using RDKit to calculate molecular descriptors from the SMILES strings of the compounds. These descriptors are numerical representations of the molecules and will be used as input features in your analysis. Common descriptors include:
    - Molecular Weight
    - LogP (octanol-water partition coefficient)
    - Topological Polar Surface Area (TPSA)
    - Number of Hydrogen Bond Donors/Acceptors
    - RDKit fingerprints (Morgan, RDKit)
4. **Exploratory Data Analysis (EDA) (Phân tích dữ liệu thăm dò):** Visualizing the data and exploring relationships between features and activity values. This is crucial for understanding the data and identifying potential patterns.
    - Scatter plots of activity vs. descriptors
    - Histograms of descriptor distributions
    - Box plots to compare activity distributions for different compound classes
5. **Modeling (Mô hình hóa):** Building predictive models to relate molecular descriptors to activity. Common approaches include:
    - Linear Regression
    - Support Vector Machines (SVM)
    - Random Forests
    - Neural Networks (for larger datasets)
6. **Model Evaluation (Đánh giá Mô hình):** Assessing the performance of your models using appropriate metrics, such as:
    - R-squared (for regression)

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Classification metrics (Accuracy, Precision, Recall, F1-score) if you're doing classification.

English and Vietnamese Code (Mã bằng tiếng Anh và tiếng Việt):

1. SQL Code (Mã SQL):

```sql
-- Topic_CheMBL_35_27.sql
-- Get 100 potent compounds (IC50, Ki, Kd < 100 nM) for a specific target. Replace
'CHEMBL25' with the actual target ID.
SELECT
    act.molregno,
    cmp.smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
WHERE
    act.target_chembl_id = 'CHEMBL25'  -- Replace with your target of interest
    AND act.standard_type IN ('IC50', 'Ki', 'Kd')
    AND act.standard_units = 'nM'
    AND act.standard_relation = '='  -- Only get exact values
    AND act.standard_value::TEXT ~ '^[0-9\.]+$' -- Make sure the standard_value is
numeric (solves error a)
    AND act.standard_value <= 100
LIMIT 100;


-- Save the results to: ../data/Topic_CheMBL_35_27.csv using pgAdmin.
```

Vietnamese:

```sql
-- Topic_CheMBL_35_27.sql
-- Lấy 100 hợp chất có hoạt tính mạnh (IC50, Ki, Kd < 100 nM) cho một mục tiêu cụ thể.
Thay 'CHEMBL25' bằng ID mục tiêu thực tế.
SELECT
    act.molregno,
    cmp.smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
WHERE
    act.target_chembl_id = 'CHEMBL25'  -- Thay thế bằng mục tiêu bạn quan tâm
    AND act.standard_type IN ('IC50', 'Ki', 'Kd')
    AND act.standard_units = 'nM'
    AND act.standard_relation = '='  -- Chỉ lấy các giá trị chính xác
    AND act.standard_value::TEXT ~ '^[0-9\.]+$' -- Đảm bảo standard_value là số (giải
quyết lỗi a)
    AND act.standard_value <= 100
LIMIT 100;
```

```
-- Lưu kết quả vào: ../data/Topic_CheMBL_35_27.csv bằng pgAdmin.
```

Explanation:

- `act.standard_value::TEXT ~ '^[0-9\.]+$'`: This line converts the `standard_value` to TEXT and checks if it contains only numbers and periods. This should resolve the `numeric ~ unknown` error.
- `act.target_chembl_id = 'CHEMBL25'`: **Important:** You *must* replace `'CHEMBL25'` with the actual ChEMBL ID of the target you're interested in. Otherwise, you'll get no data. Find this ID on the ChEMBL website.
- `LIMIT 100`: This limits the results to the first 100 rows.

2. Python Code (Mã Python):

```python
# Topic_CheMBL_35_27_1_data_processing.ipynb
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np

# Define the base path
base_path = os.path.abspath(os.path.join(os.getcwd(), "..")) # Assuming notebook is one level deep inside the project

# Construct the file path to the CSV file
csv_file_path = os.path.join(base_path, "data", "Topic_CheMBL_35_27.csv")

# Load the data
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}. Make sure the SQL query was run and the CSV file was saved correctly.")
    raise

# Drop rows with missing SMILES strings
df = df.dropna(subset=['smiles'])

# Convert standard_value to numeric, handling potential errors
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df = df.dropna(subset=['standard_value']) # Remove any rows that couldn't be converted to numeric

# Filter out molecules with invalid standard_value
df = df[df['standard_value'] > 0]

def calculate_descriptors(smiles):
    """Calculates RDKit descriptors for a given SMILES string."""
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None  # Handle invalid SMILES strings

    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    descriptors['HBD'] = Descriptors.NumHDonors(mol)
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
```

```
    descriptors['TPSA'] = Descriptors.TPSA(mol)  # Topological Polar Surface Area
    # Add more descriptors as needed
    return descriptors


# Apply the descriptor calculation function
df['descriptors'] = df['smiles'].apply(calculate_descriptors)

# Handle invalid SMILES strings
df = df.dropna(subset=['descriptors'])

# Convert the 'descriptors' column to a DataFrame
descriptors_df = pd.DataFrame(df['descriptors'].tolist())

# Concatenate the original DataFrame with the descriptors DataFrame
df = pd.concat([df, descriptors_df], axis=1)

# Drop the 'descriptors' column as it's no longer needed
df = df.drop('descriptors', axis=1)

# Display the first few rows of the DataFrame
print(df.head())

# Save the processed data to a new CSV file
output_csv_path = os.path.join(base_path, "data", "Topic_CheMBL_35_27_processed.csv")
df.to_csv(output_csv_path, index=False)

print(f"Processed data saved to {output_csv_path}")


# Topic_CheMBL_35_27_2_modeling.ipynb
import os
import pandas as pd
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

# Define the base path
base_path = os.path.abspath(os.path.join(os.getcwd(), "..")) # Assuming notebook is
one level deep inside the project

# Construct the file path to the processed CSV file
processed_csv_path = os.path.join(base_path, "data",
"Topic_CheMBL_35_27_processed.csv")

# Load the processed data
try:
    df = pd.read_csv(processed_csv_path)
except FileNotFoundError:
    print(f"Error: File not found at {processed_csv_path}. Make sure the data
processing notebook was run successfully.")
    raise

# Select features and target variable
features = ['MolWt', 'LogP', 'HBD', 'HBA', 'TPSA']  # Replace with the descriptors you
want to use
target = 'standard_value'
```

```
# Handle missing values by filling with the mean
df[features] = df[features].fillna(df[features].mean())

# Prepare the data
X = df[features]
y = df[target]

# Scale the features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the 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 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)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Example prediction for a new compound (replace with actual descriptor values)
new_compound = pd.DataFrame([[200.0, 2.5, 1, 3, 50.0]], columns=features)
new_compound_scaled = scaler.transform(new_compound)  # Scale the new data

predicted_activity = model.predict(new_compound_scaled)
print(f"Predicted activity for new compound: {predicted_activity[0]}")
```

Vietnamese:

```
# Topic_CheMBL_35_27_1_xu_ly_du_lieu.ipynb
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np

# Xác định đường dẫn gốc
base_path = os.path.abspath(os.path.join(os.getcwd(), "..")) # Giả sử notebook nằm
trong một thư mục con của dự án

# Xây dựng đường dẫn đến file CSV
csv_file_path = os.path.join(base_path, "data", "Topic_CheMBL_35_27.csv")

# Tải dữ liệu
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file tại {csv_file_path}. Hãy chắc chắn rằng bạn đã
chạy truy vấn SQL và lưu file CSV đúng cách.")
    raise
```

```python
# Loại bỏ các hàng có SMILES bị thiếu
df = df.dropna(subset=['smiles'])

# Chuyển đổi standard_value sang kiểu số, xử lý các lỗi tiềm ẩn
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df = df.dropna(subset=['standard_value'])  # Loại bỏ các hàng không thể chuyển đổi sang kiểu số

# Lọc các phân tử có standard_value không hợp lệ
df = df[df['standard_value'] > 0]


def calculate_descriptors(smiles):
    """Tính toán các descriptor RDKit cho một chuỗi SMILES."""
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None  # Xử lý các chuỗi SMILES không hợp lệ

    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    descriptors['HBD'] = Descriptors.NumHDonors(mol)
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
    descriptors['TPSA'] = Descriptors.TPSA(mol)  # Diện tích bề mặt cực topo
    # Thêm các descriptor khác nếu cần
    return descriptors


# Áp dụng hàm tính toán descriptor
df['descriptors'] = df['smiles'].apply(calculate_descriptors)

# Xử lý các chuỗi SMILES không hợp lệ
df = df.dropna(subset=['descriptors'])

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

# Kết hợp DataFrame gốc với DataFrame descriptor
df = pd.concat([df, descriptors_df], axis=1)

# Xóa cột 'descriptors' vì không còn cần thiết
df = df.drop('descriptors', axis=1)


# Hiển thị vài dòng đầu tiên của DataFrame
print(df.head())

# Lưu dữ liệu đã xử lý vào một file CSV mới
output_csv_path = os.path.join(base_path, "data", "Topic_CheMBL_35_27_processed.csv")
df.to_csv(output_csv_path, index=False)

print(f"Dữ liệu đã xử lý được lưu vào {output_csv_path}")


# Topic_CheMBL_35_27_2_mo_hinh_hoa.ipynb
import os
import pandas as pd
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

# Xác định đường dẫn gốc
base_path = os.path.abspath(os.path.join(os.getcwd(), "..")) # Giả sử notebook nằm
trong một thư mục con của dự án

# Xây dựng đường dẫn đến file CSV đã xử lý
processed_csv_path = os.path.join(base_path, "data",
"Topic_CheMBL_35_27_processed.csv")

# Tải dữ liệu đã xử lý
try:
    df = pd.read_csv(processed_csv_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file tại {processed_csv_path}. Hãy chắc chắn rằng bạn
đã chạy notebook xử lý dữ liệu thành công.")
    raise

# Chọn các đặc trưng và biến mục tiêu
features = ['MolWt', 'LogP', 'HBD', 'HBA', 'TPSA']  # Thay thế bằng các descriptor bạn
muốn sử dụng
target = 'standard_value'

# Xử lý các giá trị bị thiếu bằng cách điền vào giá trị trung bình
df[features] = df[features].fillna(df[features].mean())

# Chuẩn bị dữ liệu
X = df[features]
y = df[target]

# Chia tỷ lệ các đặc trưng bằng StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# 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)
r2 = r2_score(y_test, y_pred)

print(f"Sai số bình phương trung bình: {mse}")
print(f"Hệ số xác định R-squared: {r2}")

# Ví dụ dự đoán cho một hợp chất mới (thay thế bằng các giá trị descriptor thực tế)
new_compound = pd.DataFrame([[200.0, 2.5, 1, 3, 50.0]], columns=features)
new_compound_scaled = scaler.transform(new_compound)  # Chia tỷ lệ dữ liệu mới

predicted_activity = model.predict(new_compound_scaled)
print(f"Hoạt tính dự đoán cho hợp chất mới: {predicted_activity[0]}")
```

Explanation:

- **File Paths:** The code uses `os.path.join` to create file paths, making it more robust to different operating systems. Make sure your directory structure is correct. The `base_path` calculation assumes the notebook is in a subdirectory *below* the root of your project (where the `data` folder is located). Adjust it if necessary.
- **Error Handling:** The code includes `try...except` blocks to handle potential `FileNotFoundError` exceptions, which can occur if the CSV file is not found.
- **SMILES Handling:** The code checks for invalid SMILES strings and skips those molecules.
- **Descriptor Calculation:** The `calculate_descriptors` function calculates several common RDKit descriptors. You can easily add more descriptors.
- **Data Cleaning**: Drop missing values, remove rows where `standard_value` cannot be converted to numeric.
- **Modeling:** A simple linear regression model is used as an example. You can replace this with more complex models. Importantly, features are scaled using `StandardScaler` before training the model.
- **Model Evaluation:** The code calculates and prints the MSE and R-squared, which are common metrics for evaluating regression models.
- **Prediction Example:** The code shows how to make a prediction for a new compound, but you'll need to replace the example descriptor values with real values.
- **Split into Notebooks**: Split into two notebooks for easier processing.
    - The first notebook is for processing
    - The second notebook is for modeling

5 Examples (5 Ví dụ):

Here are five examples illustrating how you could use and extend this code:

1. **Target-Specific Analysis (Phân tích Cụ thể theo Mục tiêu):** Modify the SQL query to focus on a *different* ChEMBL target (e.g., `act.target_chembl_id = 'CHEMBL1953'`). Re-run the analysis to see how the models and key descriptors change for that target. This lets you compare structure-activity relationships across different protein targets.
    - **SQL Modification:** In the SQL code, change `'CHEMBL25'` to the desired target ID.
2. **Descriptor Selection (Lựa chọn Descriptor):** Experiment with different sets of molecular descriptors. Add more descriptors to the `calculate_descriptors` function (e.g., `Descriptors.NumRotatableBonds(mol)`, `Descriptors.RingCount(mol)`) and see which combinations lead to better model performance. Consider using feature selection techniques (e.g., SelectKBest) to automatically choose the best descriptors.
    - **Python Modification:** Add more lines within the `calculate_descriptors` function to calculate additional descriptors. Modify the `features` list in the modeling notebook to include the new descriptor names.
3. **Activity Type Filtering (Lọc Loại Hoạt tính):** Modify the SQL query to focus on a specific activity type (e.g., only `IC50` values). This can improve model accuracy if different activity types have different relationships to the descriptors.
    - **SQL Modification:** Modify the SQL `WHERE` clause to filter for a specific `standard_type`: `AND act.standard_type = 'IC50'`.
4. **Model Comparison (So sánh Mô hình):** Try different machine learning models (e.g., Random Forest, SVM) instead of linear regression. Compare their performance on the same dataset. You'll need to import the appropriate classes from `scikit-learn` and adjust the code accordingly.

    - **Python Modification:** Replace the `LinearRegression` class with another model from `scikit-learn` (e.g., `RandomForestRegressor`). Adjust the model training and prediction code accordingly.
5. **Outlier Removal (Loại bỏ Ngoại lệ):** Add code to identify and remove outliers from the data. Outliers can disproportionately influence model performance. You could use techniques like the interquartile range (IQR) method or Cook's distance to identify outliers.
    - **Python Modification:** Add code after loading the data in the first notebook to identify and remove outliers based on either the `standard_value` or the molecular descriptors.

Key Considerations:

- **Target Selection:** The choice of target (`target_chembl_id`) is *crucial*. If you don't have a good target with sufficient data, the analysis will be limited.
- **Data Quality:** Always be mindful of data quality. ChEMBL data can be noisy, and errors in the data can lead to misleading results. Careful data cleaning and validation are essential.
- **Interpretability:** Focus on models that are interpretable, meaning you can understand why they are making certain predictions. This is especially important in drug discovery, where you want to gain insights into structure-activity relationships.
- **Validation:** Rigorous validation is essential to ensure that your models generalize well to new data. Use techniques like cross-validation to assess model performance.

By following these steps and considering these points, you should be well on your way to successfully analyzing the ChEMBL 35 dataset and gaining valuable insights for drug research and development. Remember to adapt the code to your specific needs and experiment with different approaches to find what works best for your project. Good luck!