# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

Copyright 2025 | RnD Pharma Plus | www.nghiencuuthuoc.com | Zalo: +84888999311 |

## 🧩 Topic: Sequence similarity clustering of targets

Okay, I understand. You’re working on a Chembl 35 data analysis project using RDKit, following an AIMLops template, and using a PostgreSQL database. You’re facing some errors and need guidance on how to proceed.

Here’s a breakdown of the analysis model, the necessary code (SQL and Python), error fixes, file structure considerations, and examples, presented in both English and Vietnamese.

**1. Analysis Model & Strategy (English & Vietnamese)**

**English:**

The goal is to extract and analyze data from ChEMBL 35 to support drug research and development. This involves:

* **Data Extraction:** Using SQL queries to retrieve relevant data from the ChEMBL database (specifically focusing on bioactivity data). Limiting the result set to 100 rows to avoid performance issues.
* **Data Cleaning and Transformation:** Cleaning the extracted data (e.g., handling missing values, converting data types). The ERROR: operator does not exist: numeric ~ unknown indicates a type mismatch during a regular expression comparison in SQL, which needs to be addressed.
* **Feature Engineering:** Potentially generating molecular descriptors using RDKit from the ChEMBL structures (SMILES strings). This allows us to represent molecules numerically.
* **Data Analysis and Modeling:** Performing exploratory data analysis (EDA) to understand the data distribution, relationships, and potential patterns. Then, build predictive models to relate molecular properties to bioactivity. The error related to squared=False in mean\_squared\_error indicates using an older version of scikit-learn.
* **Visualization:** Visualizing the results (e.g., scatter plots, histograms) to gain insights and communicate findings.

**Vietnamese:**

Mục tiêu là trích xuất và phân tích dữ liệu từ ChEMBL 35 để hỗ trợ nghiên cứu và phát triển thuốc. Điều này bao gồm:

* **Trích xuất dữ liệu:** Sử dụng các truy vấn SQL để truy xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL (tập trung đặc biệt vào dữ liệu hoạt tính sinh học). Giới hạn tập kết quả thành 100 hàng để tránh các vấn đề về hiệu suất.
* **Làm sạch và chuyển đổi dữ liệu:** Làm sạch dữ liệu đã trích xuất (ví dụ: xử lý các giá trị bị thiếu, chuyển đổi các loại dữ liệu). Lỗi ERROR: operator does not exist: numeric ~ unknown cho thấy sự không khớp loại trong quá trình so sánh biểu thức chính quy trong SQL, cần được giải quyết.
* **Kỹ thuật đặc trưng:** Có khả năng tạo ra các mô tả phân tử bằng cách sử dụng RDKit từ cấu trúc ChEMBL (chuỗi SMILES). Điều này cho phép chúng ta biểu diễn các phân tử bằng số.
* **Phân tích và mô hình hóa dữ liệu:** Thực hiện phân tích dữ liệu thăm dò (EDA) để hiểu phân phối dữ liệu, mối quan hệ và các mẫu tiềm năng. Sau đó, xây dựng các mô hình dự đoán để liên hệ các thuộc tính phân tử với hoạt tính sinh học. Lỗi liên quan đến squared=False trong mean\_squared\_error cho thấy việc sử dụng phiên bản scikit-learn cũ hơn.
* **Trực quan hóa:** Trực quan hóa kết quả (ví dụ: biểu đồ phân tán, biểu đồ tần suất) để có được thông tin chi tiết và truyền đạt kết quả.

**2. Code (SQL & Python) - English & Vietnamese**

**SQL (English):**

-- File: ../data/chembl\_bioactivity\_100.csv  
  
SELECT  
 cmp.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 act.activity\_comment,  
 mol.molfile  
FROM  
 activities act  
JOIN  
 molecule\_dictionary cmp ON act.molregno = cmp.molregno  
JOIN  
 compound\_structures mol ON cmp.molregno = mol.molregno  
WHERE  
 act.standard\_type = 'IC50' -- Filter for a specific activity type  
 AND act.standard\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9\.]+$' --Fix:Cast standard\_value to text before applying regex  
 AND act.standard\_units = 'nM'  
LIMIT 100;

**SQL (Vietnamese):**

-- File: ../data/chembl\_bioactivity\_100.csv  
  
SELECT  
 cmp.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 act.activity\_comment,  
 mol.molfile  
FROM  
 activities act  
JOIN  
 molecule\_dictionary cmp ON act.molregno = cmp.molregno  
JOIN  
 compound\_structures mol ON cmp.molregno = mol.molregno  
WHERE  
 act.standard\_type = 'IC50' -- Lọc cho một loại hoạt động cụ thể  
 AND act.standard\_relation = '='  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9\.]+$' -- Sửa lỗi: Ép standard\_value thành text trước khi áp dụng biểu thức chính quy  
 AND act.standard\_units = 'nM'  
LIMIT 100;

**Explanation:**

* The SQL query selects relevant information (ChEMBL ID, activity type, value, units, activity comment, and molfile) from the activities, molecule\_dictionary, and compound\_structures tables.
* The WHERE clause filters for:
  + standard\_type = 'IC50': IC50 values (a common measure of bioactivity). You can change this to another activity type if needed.
  + standard\_relation = '=': Only exact values, not greater than or less than.
  + standard\_value IS NOT NULL: Excludes rows with missing activity values.
  + act.standard\_value::text ~ '^[0-9\.]+$': Ensures that the standard\_value contains only numbers and periods (to filter out invalid entries). This fixes the “operator does not exist” error by explicitly casting the numeric standard\_value to text before applying the regular expression.
  + standard\_units = 'nM': Values in nanomolar (nM).
* LIMIT 100 limits the result set to 100 rows. This is crucial for your memory constraints.

**Important:**

1. **Run this SQL code using pgAdmin** on your PostgreSQL server (192.168.206.136, user: rd, pass: rd, database: chembl\_35).
2. **Save the results** as a CSV file named chembl\_bioactivity\_100.csv in the ../data/ directory of your AIMLops project. Make sure the CSV file has a header row.

**Python (English):**

# File: notebook/Topic\_CheMBL\_35\_33\_1\_data\_analysis.ipynb  
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  
import matplotlib.pyplot as plt  
  
  
# Define the base path  
base\_path = ".." # Assuming notebook is in the notebook directory  
  
# Construct the path to the CSV file  
data\_file = os.path.join(base\_path, "data", "chembl\_bioactivity\_100.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 file.")  
 exit()  
  
# Data Cleaning and Preprocessing  
df = df.dropna(subset=['molfile', 'standard\_value']) # Remove rows with missing SMILES or activity values  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce') # ensure correct type and drop errors  
df = df.dropna(subset=['standard\_value']) # Remove rows where conversion to numeric failed  
df = df[df['standard\_value'] > 0] # Remove non-positive values (log transform will fail)  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Convert IC50 to pIC50  
  
# Feature Engineering (Molecular Descriptors)  
def calculate\_descriptors(mol):  
 try:  
 descriptors = {}  
 descriptors['MW'] = Descriptors.MolWt(mol)  
 descriptors['LogP'] = Descriptors.MolLogP(mol)  
 descriptors['HBA'] = Descriptors.NumHAcceptors(mol)  
 descriptors['HBD'] = Descriptors.NumHDonors(mol)  
 descriptors['TPSA'] = Descriptors.TPSA(mol)  
  
 return pd.Series(descriptors)  
 except:  
 return pd.Series([None]\*5) # handle exceptions  
  
mols = [Chem.MolFromMolBlock(mol) for mol in df['molfile']]  
df[['MW', 'LogP', 'HBA', 'HBD', 'TPSA']] = pd.DataFrame([calculate\_descriptors(mol) if mol else [None]\*5 for mol in mols])  
  
df = df.dropna() # Drop any rows with NA from descriptor calculation  
  
  
# Model Training  
X = df[['MW', 'LogP', 'HBA', 'HBD', 'TPSA']]  
y = df['pIC50']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Model Evaluation  
y\_pred = model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred) # No need for squared=False, using a current scikit-learn version is preferable  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")  
  
# Visualization  
plt.scatter(y\_test, y\_pred)  
plt.xlabel("Actual pIC50")  
plt.ylabel("Predicted pIC50")  
plt.title("Actual vs. Predicted pIC50")  
plt.show()

**Python (Vietnamese):**

# File: notebook/Topic\_CheMBL\_35\_33\_1\_phan\_tich\_du\_lieu.ipynb  
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  
import matplotlib.pyplot as plt  
  
  
# Định nghĩa đường dẫn cơ sở  
base\_path = ".." # Giả sử notebook nằm trong thư mục notebook  
  
# Xây dựng đường dẫn đến tệp CSV  
data\_file = os.path.join(base\_path, "data", "chembl\_bioactivity\_100.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 tệp CSV.")  
 exit()  
  
# Làm sạch và tiền xử lý dữ liệu  
df = df.dropna(subset=['molfile', 'standard\_value']) # Xóa các hàng có SMILES hoặc giá trị hoạt động bị thiếu  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce') # đảm bảo kiểu chính xác và loại bỏ lỗi  
df = df.dropna(subset=['standard\_value']) # Xóa các hàng mà chuyển đổi sang số không thành công  
df = df[df['standard\_value'] > 0] # Loại bỏ các giá trị không dương (chuyển đổi log sẽ thất bại)  
df['pIC50'] = -np.log10(df['standard\_value'] \* 1e-9) # Chuyển đổi IC50 sang pIC50  
  
# Kỹ thuật đặc trưng (Mô tả phân tử)  
def calculate\_descriptors(mol):  
 try:  
 descriptors = {}  
 descriptors['MW'] = Descriptors.MolWt(mol)  
 descriptors['LogP'] = Descriptors.MolLogP(mol)  
 descriptors['HBA'] = Descriptors.NumHAcceptors(mol)  
 descriptors['HBD'] = Descriptors.NumHDonors(mol)  
 descriptors['TPSA'] = Descriptors.TPSA(mol)  
  
 return pd.Series(descriptors)  
 except:  
 return pd.Series([None]\*5) # Xử lý ngoại lệ  
  
mols = [Chem.MolFromMolBlock(mol) for mol in df['molfile']]  
df[['MW', 'LogP', 'HBA', 'HBD', 'TPSA']] = pd.DataFrame([calculate\_descriptors(mol) if mol else [None]\*5 for mol in mols])  
  
df = df.dropna() # Loại bỏ bất kỳ hàng nào có NA từ tính toán mô tả  
  
  
# Huấn luyện mô hình  
X = df[['MW', 'LogP', 'HBA', 'HBD', 'TPSA']]  
y = df['pIC50']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
# Đánh giá mô hình  
y\_pred = model.predict(X\_test)  
mse = mean\_squared\_error(y\_test, y\_pred) # Không cần squared=False, tốt nhất là sử dụng phiên bản scikit-learn hiện tại  
r2 = r2\_score(y\_test, y\_pred)  
  
print(f"Lỗi bình phương trung bình: {mse}")  
print(f"R-squared: {r2}")  
  
# Trực quan hóa  
plt.scatter(y\_test, y\_pred)  
plt.xlabel("pIC50 thực tế")  
plt.ylabel("pIC50 dự đoán")  
plt.title("pIC50 thực tế so với pIC50 dự đoán")  
plt.show()

**Explanation:**

1. **Import Libraries:** Imports necessary libraries (pandas, RDKit, scikit-learn, matplotlib).
2. **File Path:** Uses os.path.join(base\_path, "data", "chembl\_bioactivity\_100.csv") to construct the correct file path to your data.
3. **Data Loading:** Loads the CSV data into a pandas DataFrame. Includes error handling if the file is not found.
4. **Data Cleaning:**
   * Removes rows with missing molfile or standard\_value.
   * Converts standard\_value to numeric and removes rows where the conversion fails.
   * Removes non-positive standard values, as log transformation is used later.
   * Converts IC50 values to pIC50 (a more common representation in drug discovery). The conversion is pIC50 = -log10(IC50 in Molar). Since your IC50 is in nM, we multiply by 1e-9 to convert to Molar.
5. **Feature Engineering:**
   * Defines a function calculate\_descriptors that uses RDKit to calculate common molecular descriptors (Molecular Weight, LogP, Hydrogen Bond Acceptors, Hydrogen Bond Donors, and Topological Polar Surface Area).
   * Iterates through the molecules and calculates the descriptors. Handles potential errors during descriptor calculation.
   * Creates new columns in the DataFrame for the calculated descriptors.
   * Drops any remaining rows with NA values that may have resulted from descriptor calculation issues.
6. **Model Training:**
   * Selects the molecular descriptors as features (X) and pIC50 as the target variable (y).
   * Splits the data into training and testing sets.
   * Trains a linear regression model.
7. **Model Evaluation:**
   * Predicts pIC50 values for the test set.
   * Calculates the Mean Squared Error (MSE) and R-squared (R2) to evaluate the model’s performance.
   * Prints the evaluation metrics.
8. **Visualization:** Creates a scatter plot of actual vs. predicted pIC50 values.

**Important Considerations and Error Fixes:**

* **File Paths:** Double-check that the file paths are correct. The base\_path = ".." assumes your notebook is in a subdirectory called “notebook” and that the data is in the ../data directory. Adjust this if your file structure is different.
* **RDKit Installation:** Make sure you have RDKit installed in your Jupyter Notebook environment. You can install it using conda install -c conda-forge rdkit or pip install rdkit.
* **scikit-learn Version:** The squared=False error in mean\_squared\_error indicates you are using an older version of scikit-learn. The best approach is to update scikit-learn: conda update scikit-learn or pip install --upgrade scikit-learn. However, the code above removes the squared=False parameter to maintain compatibility with older versions.
* **Data Cleaning is Crucial:** The quality of your model depends on the quality of your data. Pay close attention to handling missing values, incorrect data types, and outliers.
* \*\* molfile must be molblock format:\*\* The rdkit reads the molfile from the database in the format molblock, otherwise it can read the smiles format

**3. AIMLops Folder Tree & File Placement**

Based on the AIMLops template, your structure should look something like this:

Topic\_CheMBL\_35\_33/  
├── data/  
│ └── chembl\_bioactivity\_100.csv # Your extracted data  
├── notebooks/  
│ └── Topic\_CheMBL\_35\_33\_1\_data\_analysis.ipynb # Jupyter Notebook  
├── src/  
│ └── # (Optional: For more complex code, refactor functions into Python modules here)  
└── models/  
 └── # (Optional: Save trained models here)

**4. 5 Examples of How to Use the Code**

Here are 5 ways you can adapt and use the provided code:

1. **Change the Activity Type:** Modify the SQL query to extract data for a different standard\_type (e.g., 'Ki', 'Kd'). Then, adjust the interpretation in the Python code accordingly.

* WHERE  
   act.standard\_type = 'Ki' -- Example: Change to Ki  
   AND act.standard\_relation = '='  
   AND act.standard\_value IS NOT NULL  
   AND act.standard\_value::text ~ '^[0-9\.]+$'  
   AND act.standard\_units = 'nM'

1. **Explore Different Descriptors:** Add or remove molecular descriptors in the calculate\_descriptors function. RDKit provides a wide range of descriptors. Refer to the RDKit documentation for available descriptors.

* def calculate\_descriptors(mol):  
   try:  
   descriptors = {}  
   descriptors['MW'] = Descriptors.MolWt(mol)  
   descriptors['LogP'] = Descriptors.MolLogP(mol)  
   descriptors['HBA'] = Descriptors.NumHAcceptors(mol)  
   descriptors['HBD'] = Descriptors.NumHDonors(mol)  
   descriptors['TPSA'] = Descriptors.TPSA(mol)  
   descriptors['RotatableBonds'] = Descriptors.NumRotatableBonds(mol) # Example: Add Rotatable Bonds  
   return pd.Series(descriptors)  
   except:  
   return pd.Series([None]\*6) # handle exceptions  
    
  mols = [Chem.MolFromMolBlock(mol) for mol in df['molfile']]  
  df[['MW', 'LogP', 'HBA', 'HBD', 'TPSA','RotatableBonds']] = pd.DataFrame([calculate\_descriptors(mol) if mol else [None]\*6 for mol in mols])

1. **Try a Different Model:** Experiment with different machine learning models from scikit-learn (e.g., Random Forest, Support Vector Regression). You’ll need to adjust the model instantiation and fitting steps.

* from sklearn.ensemble import RandomForestRegressor  
    
  model = RandomForestRegressor(n\_estimators=100, random\_state=42) # Example: Random Forest  
  model.fit(X\_train, y\_train)

1. **Adjust the Test Size:** Change the test\_size parameter in train\_test\_split to control the proportion of data used for testing.

* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42) # Example: 30% for testing

1. **Add More Data:** If you have more computational resources, increase the LIMIT in the SQL query to retrieve more data. Be mindful of memory usage. You might need to use techniques like batch processing if you are dealing with very large datasets.

* LIMIT 1000; -- Example: Get 1000 rows (be cautious!)

Remember to carefully analyze the results of each modification to understand its impact on the model’s performance. Good luck!