# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Correlation heatmap of physicochemical properties

Okay, I understand. You’re working with Chembl 35 data using RDKit within an AIMLops framework, and you’re encountering a few issues. I’ll help you analyze the problem, provide corrected code examples, and offer guidance in both English and Vietnamese.

**Analysis**

1. **Analysis Model Overview**

* Your project likely involves these key steps:
  + **Data Extraction (SQL):** Querying the Chembl 35 database to extract relevant bioactivity data (activity values, target information, compound structures - SMILES). You’ll save this data to CSV files.
  + **Data Preprocessing (Python/RDKit):**
    - Reading the CSV data into a Pandas DataFrame.
    - Cleaning and filtering the data (handling missing values, removing invalid structures).
    - Converting SMILES strings to RDKit Mol objects.
    - Calculating molecular descriptors (features) using RDKit.
  + **Exploratory Data Analysis (EDA):** Visualizing the data to understand distributions, correlations, and identify potential patterns.
  + **Model Building (Python/Scikit-learn):**
    - Splitting the data into training and test sets.
    - Training a machine learning model (e.g., Linear Regression, Random Forest) to predict bioactivity based on the calculated descriptors.
    - Evaluating the model’s performance on the test set using metrics like Mean Squared Error (MSE), R-squared.
  + **Interpretation and Reporting:** Analyzing the model’s coefficients/feature importance to gain insights into the structure-activity relationship (SAR).

1. **Error Analysis**
   * **Error (a): ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard\_value ~ '^[0-9\.]+$'**
   * This error occurs because you’re trying to use the regular expression operator ~ (for string matching) on a numeric column (act.standard\_value). PostgreSQL likely has act.standard\_value defined as a numeric type. You need to either cast the column to text or use numeric comparison operators.
   * **Error (b): old scikit-learn version does not support parameters squared=False in the mean\_squared\_error function**
   * This means you have an older version of scikit-learn installed. The squared=False argument was added to mean\_squared\_error in a later version. You have two options:
     + **Upgrade scikit-learn:** This is the recommended approach. pip install -U scikit-learn
     + **Calculate RMSE manually:** If upgrading isn’t feasible, you can calculate the Root Mean Squared Error (RMSE) by taking the square root of the MSE.

**Code Examples (English & Vietnamese)**

**Directory Structure (AIMLops)**

Topic\_CheMBL\_35\_77/  
├── data/  
│ └── chembl\_activity\_data.csv # Your extracted data will be saved here  
├── notebooks/  
│ └── Topic\_CheMBL\_35\_77\_1\_Data\_Extraction\_Preprocessing.ipynb  
│ └── Topic\_CheMBL\_35\_77\_2\_Model\_Building\_Evaluation.ipynb  
├── src/  
│ └── utils.py # Optional: For reusable functions  
└── README.md

**1. SQL Code (Extracting Data - Limit 100 rows)**

-- Topic\_CheMBL\_35\_77\_data\_extraction.sql  
  
SELECT  
 act.activity\_id,  
 cmp.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 act.pchembl\_value,  
 md.molregno,  
 cs.canonical\_smiles  
FROM  
 activities act  
JOIN  
 molecule\_dictionary md ON act.molregno = md.molregno  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
WHERE  
 act.standard\_type = 'IC50' -- Example: Filter for IC50 values  
 AND act.standard\_units = 'nM' -- Example: Filter for nM units  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9\.]+$' -- Ensure standard\_value contains only numbers and dots  
LIMIT 100;

**Vietnamese:**

-- Topic\_CheMBL\_35\_77\_data\_extraction.sql  
  
SELECT  
 act.activity\_id,  
 cmp.chembl\_id,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 act.pchembl\_value,  
 md.molregno,  
 cs.canonical\_smiles  
FROM  
 activities act  
JOIN  
 molecule\_dictionary md ON act.molregno = md.molregno  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
WHERE  
 act.standard\_type = 'IC50' -- Ví dụ: Lọc các giá trị IC50  
 AND act.standard\_units = 'nM' -- Ví dụ: Lọc các đơn vị nM  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9\.]+$' -- Đảm bảo standard\_value chỉ chứa số và dấu chấm  
LIMIT 100;

**Explanation:**

* This SQL query joins the activities, molecule\_dictionary, and compound\_structures tables to retrieve activity data, compound IDs, and SMILES strings.
* It filters for IC50 values in nM units. Adjust these filters as needed for your specific research question.
* The important change is to add ::text to cast numeric field to text so that we can use the Regular Expression

**2. Python Code (Notebook 1: Data Extraction and Preprocessing)**

# Topic\_CheMBL\_35\_77\_1\_Data\_Extraction\_Preprocessing.ipynb  
  
import pandas as pd  
import os  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
# Define base path for file operations  
base\_path = os.getcwd() # Gets the current working directory  
data\_path = os.path.join(base\_path, 'data', 'chembl\_activity\_data.csv')  
  
# Load the data from CSV  
try:  
 df = pd.read\_csv(data\_path)  
except FileNotFoundError:  
 print(f"Error: File not found at {data\_path}. Make sure you've run the SQL query and saved the data.")  
 exit()  
  
print(f"Data loaded. Shape: {df.shape}")  
  
# Data Cleaning and Preprocessing  
df = df.dropna(subset=['canonical\_smiles', 'standard\_value']) #Drop all the raw without SMILES and standard\_value  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce') #Coerce error values to NaN  
df = df.dropna(subset=['standard\_value']) #Drop all the raws without standard\_value  
  
# RDKit Processing: Create Mol objects and calculate descriptors  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol)] # Example Descriptors  
 else:  
 return None  
  
df['descriptors'] = df['canonical\_smiles'].apply(calculate\_descriptors)  
df = df.dropna(subset=['descriptors']) # Remove rows where descriptor calculation failed  
  
# Split descriptors into separate columns  
df[['mol\_weight', 'logP']] = pd.DataFrame(df['descriptors'].tolist(), index=df.index)  
df = df.drop('descriptors', axis=1)  
  
print(df.head())  
print(df.shape)

**Vietnamese:**

# Topic\_CheMBL\_35\_77\_1\_Data\_Extraction\_Preprocessing.ipynb  
  
import pandas as pd  
import os  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
# Xác định đường dẫn gốc cho các thao tác với file  
base\_path = os.getcwd() # Lấy thư mục làm việc hiện tại  
data\_path = os.path.join(base\_path, 'data', 'chembl\_activity\_data.csv')  
  
# Tải dữ liệu từ file CSV  
try:  
 df = pd.read\_csv(data\_path)  
except FileNotFoundError:  
 print(f"Lỗi: Không tìm thấy file tại {data\_path}. Đảm bảo bạn đã chạy truy vấn SQL và lưu dữ liệu.")  
 exit()  
  
print(f"Dữ liệu đã được tải. Shape: {df.shape}")  
  
# Làm sạch và tiền xử lý dữ liệu  
df = df.dropna(subset=['canonical\_smiles', 'standard\_value']) #Loại bỏ các dòng thiếu SMILES và standard\_value  
df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce') #Chuyển đổi lỗi sang NaN  
df = df.dropna(subset=['standard\_value']) #Loại bỏ các dòng thiếu standard\_value  
  
# Xử lý bằng RDKit: Tạo đối tượng Mol và tính toán các descriptor  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol)] # Ví dụ về các descriptor  
 else:  
 return None  
  
df['descriptors'] = df['canonical\_smiles'].apply(calculate\_descriptors)  
df = df.dropna(subset=['descriptors']) # Loại bỏ các hàng tính toán descriptor bị lỗi  
  
# Tách các descriptor thành các cột riêng biệt  
df[['mol\_weight', 'logP']] = pd.DataFrame(df['descriptors'].tolist(), index=df.index)  
df = df.drop('descriptors', axis=1)  
  
print(df.head())  
print(df.shape)

**Explanation:**

* Loads the data from the CSV file. Handles the FileNotFoundError gracefully.
* Cleans the data by dropping rows with missing SMILES or standard values.
* Defines a function calculate\_descriptors to calculate molecular descriptors using RDKit. **Important:** This example calculates molecular weight and LogP. You should choose descriptors relevant to your research question.
* Applies the calculate\_descriptors function to the SMILES strings.
* Splits the list of descriptors into individual columns.
* Prints the head of the dataframe and the shape.

**3. Python Code (Notebook 2: Model Building and Evaluation)**

# Topic\_CheMBL\_35\_77\_2\_Model\_Building\_Evaluation.ipynb  
  
import pandas as pd  
import os  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
import numpy as np  
  
# Define base path for file operations  
base\_path = os.getcwd() # Gets the current working directory  
data\_path = os.path.join(base\_path, 'data', 'chembl\_activity\_data.csv')  
  
# Load the preprocessed data (assuming you've run the previous notebook)  
try:  
 df = pd.read\_csv(data\_path)  
except FileNotFoundError:  
 print(f"Error: File not found at {data\_path}. Make sure you've run the previous notebook.")  
 exit()  
  
# Prepare data for modeling  
X = df[['mol\_weight', 'logP']] # Features  
y = df['standard\_value'] #Target  
  
# 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 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}")  
  
#If older version of scikit-learn, calculate RMSE manually:  
rmse = np.sqrt(mse)  
print(f"Root Mean Squared Error: {rmse}")

**Vietnamese:**

# Topic\_CheMBL\_35\_77\_2\_Model\_Building\_Evaluation.ipynb  
  
import pandas as pd  
import os  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
import numpy as np  
  
# Xác định đường dẫn gốc cho các thao tác với file  
base\_path = os.getcwd() # Lấy thư mục làm việc hiện tại  
data\_path = os.path.join(base\_path, 'data', 'chembl\_activity\_data.csv')  
  
# Tải dữ liệu đã được tiền xử lý (giả sử bạn đã chạy notebook trước đó)  
try:  
 df = pd.read\_csv(data\_path)  
except FileNotFoundError:  
 print(f"Lỗi: Không tìm thấy file tại {data\_path}. Đảm bảo bạn đã chạy notebook trước đó.")  
 exit()  
  
# Chuẩn bị dữ liệu cho mô hình  
X = df[['mol\_weight', 'logP']] # Các features  
y = df['standard\_value'] # 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)  
  
# Huấn luyện mô hình Linear Regression  
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"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")  
  
#Nếu phiên bản scikit-learn cũ, tính toán RMSE thủ công:  
rmse = np.sqrt(mse)  
print(f"Root Mean Squared Error: {rmse}")

**Explanation:**

* Loads the preprocessed data from the CSV file (created in the first notebook).
* Selects the molecular weight and LogP as features (X) and the standard value as the target variable (y). **Important:** You’ll likely want to use more features (more descriptors) for a better model.
* Splits the data into training and testing sets.
* Trains a Linear Regression model. You can experiment with other models like Random Forest.
* Makes predictions on the test set.
* Calculates and prints the Mean Squared Error (MSE), R-squared, and Root Mean Squared Error (RMSE).

**4. Examples**

Here are five examples demonstrating how you can modify the code:

1. **Changing Descriptors:**

* # In Notebook 1, modify the calculate\_descriptors function:  
  def calculate\_descriptors(smiles):  
   mol = Chem.MolFromSmiles(smiles)  
   if mol is not None:  
   return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol), Descriptors.NumHAcceptors(mol), Descriptors.NumHDonors(mol)] # Added two more descriptors  
   else:  
   return None  
    
  # Update the column selection in Notebook 2  
  X = df[['mol\_weight', 'logP', 'NumHAcceptors', 'NumHDonors']]
* **Vietnamese:**
* # Trong Notebook 1, sửa đổi hàm calculate\_descriptors:  
  def calculate\_descriptors(smiles):  
   mol = Chem.MolFromSmiles(smiles)  
   if mol is not None:  
   return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol), Descriptors.NumHAcceptors(mol), Descriptors.NumHDonors(mol)] # Thêm hai descriptor  
   else:  
   return None  
    
  # Cập nhật lựa chọn cột trong Notebook 2  
  X = df[['mol\_weight', 'logP', 'NumHAcceptors', 'NumHDonors']]

1. **Trying a Different Model (Random Forest):**

* # In Notebook 2, replace the Linear Regression model:  
  from sklearn.ensemble import RandomForestRegressor  
    
  model = RandomForestRegressor(n\_estimators=100, random\_state=42) # Example parameters  
  model.fit(X\_train, y\_train)
* **Vietnamese:**
* # Trong Notebook 2, thay thế mô hình Linear Regression:  
  from sklearn.ensemble import RandomForestRegressor  
    
  model = RandomForestRegressor(n\_estimators=100, random\_state=42) # Ví dụ về các tham số  
  model.fit(X\_train, y\_train)

1. **Transforming the Target Variable (Log Transformation):** Sometimes, taking the logarithm of the activity values can improve the model’s performance.

* # In Notebook 1 (after loading the data):  
  import numpy as np  
  df['standard\_value'] = np.log10(df['standard\_value'])  
    
  # Remember to transform predictions back when evaluating in Notebook 2:  
  y\_pred = model.predict(X\_test)  
  y\_pred\_original = 10\*\*y\_pred #Transform to the original value before evaluate the result.  
  mse = mean\_squared\_error(y\_test, y\_pred\_original)  
  r2 = r2\_score(y\_test, y\_pred\_original)
* **Vietnamese:**
* # Trong Notebook 1 (sau khi tải dữ liệu):  
  import numpy as np  
  df['standard\_value'] = np.log10(df['standard\_value'])  
    
  # Nhớ chuyển đổi dự đoán trở lại khi đánh giá trong Notebook 2:  
  y\_pred = model.predict(X\_test)  
  y\_pred\_original = 10\*\*y\_pred #Chuyển về giá trị ban đầu trước khi đánh giá kết quả  
  mse = mean\_squared\_error(y\_test, y\_pred\_original)  
  r2 = r2\_score(y\_test, y\_pred\_original)

1. **Adding More Filters in SQL:**

* -- In your SQL query:  
  AND act.pchembl\_value IS NOT NULL -- Filter for compounds with pChEMBL values  
  AND md.number\_of\_rings > 0 -- Filter for compounds with at least one ring
* **Vietnamese:**
* -- Trong truy vấn SQL của bạn:  
  AND act.pchembl\_value IS NOT NULL -- Lọc các hợp chất có giá trị pChEMBL  
  AND md.number\_of\_rings > 0 -- Lọc các hợp chất có ít nhất một vòng

1. **Using different standard type**

* -- In your SQL query:  
   WHERE  
   act.standard\_type = 'Ki' -- Example: Filter for Ki values  
   AND act.standard\_units = 'nM' -- Example: Filter for nM units  
   AND act.standard\_value IS NOT NULL  
   AND act.standard\_value::text ~ '^[0-9\.]+$' -- Ensure standard\_value contains only numbers and dots  
   LIMIT 100;  
   ```  
    
  \*\*Vietnamese:\*\*  
    
  ```sql  
  -- Trong truy vấn SQL của bạn:  
  WHERE  
   act.standard\_type = 'Ki' -- Ví dụ: Lọc các giá trị Ki  
   AND act.standard\_units = 'nM' -- Ví dụ: Lọc các đơn vị nM  
   AND act.standard\_value IS NOT NULL  
   AND act.standard\_value::text ~ '^[0-9\.]+$' -- Đảm bảo standard\_value chỉ chứa số và dấu chấm  
  LIMIT 100;

**Key Improvements and Considerations:**

* **Error Handling:** The Python code includes basic error handling (e.g., checking for FileNotFoundError). Add more robust error handling as needed.
* **Feature Selection:** Carefully consider which molecular descriptors to use as features. Domain knowledge is crucial here. Experiment with different feature sets.
* **Model Selection:** Don’t just rely on Linear Regression. Try other machine learning models (e.g., Random Forest, Support Vector Machines).
* **Hyperparameter Tuning:** Optimize the hyperparameters of your chosen model using techniques like grid search or random search.
* **Data Scaling/Normalization:** Consider scaling or normalizing your features, especially if you’re using models that are sensitive to feature scaling (e.g., Support Vector Machines, K-Nearest Neighbors).
* **Cross-Validation:** Use cross-validation to get a more reliable estimate of your model’s performance.
* **Units:** Be extremely careful about units. Ensure that all your activity values are in the same units.
* **Database Connection:** If you need to interact with the database directly from Python (instead of exporting to CSV), use a library like psycopg2.
* **Chembl Version:** Be aware of the specific differences between Chembl versions. The structure of the database can change.

This comprehensive guide should get you started. Remember to adapt the code to your specific research question and data. Good luck!