# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Extracting lead-like compounds

Okay, I understand. You’re working on a Chembl 35 data analysis project using RDKit, following an AIMLops template, and running into a few specific issues. I’ll help you address these, provide SQL and Python code snippets, and create 5 examples to get you started. I will provide both English and Chinese explanations where necessary.

Here’s the plan:

**1. Analysis of the Analysis Model:**

The general goal is likely to extract relevant chemical and biological activity data from the Chembl 35 database, process it using RDKit for feature generation (molecular descriptors, fingerprints), and then apply machine learning models to predict activity or understand structure-activity relationships (SAR). Since you mentioned mean\_squared\_error, it suggests you’re focusing on a regression problem – predicting a continuous activity value (like IC50, Ki, etc.).

Here’s a breakdown of the likely steps:

* **Data Extraction (SQL):** Query the Chembl 35 database to retrieve relevant tables (activities, molecules, compound\_structures, etc.) and filter for specific criteria (e.g., assay types, target families, activity types). A key aspect is filtering the standard\_value to include only numeric values for regression.
* **Data Processing (Python/RDKit):** Load the extracted data into a Pandas DataFrame. Clean the data (handle missing values). Use RDKit to generate molecular descriptors and fingerprints from the SMILES strings.
* **Model Building (Python/Scikit-learn):** Split the data into training and testing sets. Choose a regression model (e.g., Linear Regression, Random Forest Regressor, Gradient Boosting Regressor). Train the model on the training data. Evaluate the model on the testing data using metrics like Mean Squared Error (MSE), R-squared.
* **Analysis and Interpretation:** Analyze the model’s performance, identify important features, and interpret the SAR.

**2. Code (SQL & Python):**

**2.1 SQL (to be run in pgAdmin, then saved to ../data/Topic\_CheMBL\_35\_3.csv):**

-- Topic\_CheMBL\_35\_3.sql  
-- Extract data from ChEMBL 35 for activity prediction  
  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 act.assay\_id  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 activities act ON md.molregno = act.molregno  
WHERE  
 act.standard\_type = 'IC50' -- Or other relevant activity type  
 AND act.standard\_units = 'nM' -- Or other relevant units  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9.]+$' --Fix for numeric ~ unknown error, using text casting and regex. This is the crucial fix.  
 AND length(cs.canonical\_smiles) < 200 --Avoid very large molecules for initial testing  
LIMIT 100; -- Limit to 100 rows for testing

**Explanation (English):**

* This SQL query retrieves data from molecule\_dictionary, compound\_structures, and activities tables.
* It filters for records where standard\_type is ‘IC50’ and standard\_units is ‘nM’. You can adjust these based on your target.
* act.standard\_value IS NOT NULL ensures we’re not including rows with missing values.
* act.standard\_value::text ~ '^[0-9.]+$' **This is the fix for your numeric ~ unknown error.** It casts the standard\_value to text and then uses a regular expression to ensure it contains only numbers and periods (to avoid values like “>1000” or “<10”). This approach is much more robust than directly comparing against numeric types in the presence of potentially non-numeric data.
* LIMIT 100 limits the output to 100 rows for faster testing.

**Explanation (Chinese):**

* 此SQL查询从molecule\_dictionary，compound\_structures和activities表中检索数据。
* 它过滤了standard\_type为’IC50’和standard\_units为’nM’的记录。 您可以根据您的目标调整这些。
* act.standard\_value IS NOT NULL 确保我们不包括缺少值的行。
* act.standard\_value::text ~ '^[0-9.]+$' **这是解决您的numeric ~ unknown错误的修复方案。** 它将standard\_value转换为文本，然后使用正则表达式来确保它仅包含数字和句点（以避免诸如”>1000”或”<10”之类的值）。 这种方法比在存在潜在非数字数据的情况下直接与数字类型进行比较更强大。
* LIMIT 100 将输出限制为100行，以加快测试速度。

**2.2 Python (Jupyter Notebook, notebook/Topic\_CheMBL\_35\_3\_1\_DataPrep.ipynb):**

# Topic\_CheMBL\_35\_3\_1\_DataPrep.ipynb  
import os  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import AllChem  
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  
from sklearn.ensemble import RandomForestRegressor #Example model  
import warnings  
warnings.filterwarnings("ignore")  
  
# Define the base path  
base\_path = os.path.abspath(os.path.join(os.getcwd(), "..")) # Assuming notebook is in /notebook  
  
# Construct the data path  
data\_path = os.path.join(base\_path, "data", "Topic\_CheMBL\_35\_3.csv")  
print(f"Loading data from: {data\_path}")  
  
# Load the data  
try:  
 df = pd.read\_csv(data\_path)  
except FileNotFoundError:  
 print(f"Error: File not found at {data\_path}. Make sure you ran the SQL script and saved the CSV.")  
 exit()  
  
# Data Cleaning and Preprocessing  
print("Original data shape:", df.shape)  
df = df.dropna(subset=['canonical\_smiles', 'standard\_value']) # Drop rows with missing SMILES or activity values  
df = df[pd.to\_numeric(df['standard\_value'], errors='coerce').notna()] #Ensure Standard value is numeric  
df['standard\_value'] = pd.to\_numeric(df['standard\_value']) #Convert standard value to numeric  
print("Cleaned data shape:", df.shape)  
  
# RDKit Feature Generation (Example: Morgan Fingerprints)  
def generate\_morgan\_fingerprint(smiles, radius=2, nBits=2048):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits)  
 return np.array(list(fp))  
 else:  
 return None  
 except:  
 return None #Handle parsing errors  
  
df['morgan\_fp'] = df['canonical\_smiles'].apply(generate\_morgan\_fingerprint)  
df = df.dropna(subset=['morgan\_fp']) #Remove rows where fingerprint generation failed.  
print("Data shape after fingerprint generation:", df.shape)  
  
# Model Training (Example: Linear Regression)  
X = np.vstack(df['morgan\_fp'].to\_numpy())  
y = np.log10(df['standard\_value'].to\_numpy()) # Log transform activity for better distribution  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) #Random state for reproducibility  
  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
y\_pred = model.predict(X\_test)  
  
# Evaluate the model  
mse = mean\_squared\_error(y\_test, y\_pred) #Older sklearn versions don't need squared=False  
r2 = r2\_score(y\_test, y\_pred)  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2}")  
  
#Example with Random Forest Regression  
model\_rf = RandomForestRegressor(n\_estimators=100, random\_state=42) # You can adjust hyperparameters  
model\_rf.fit(X\_train, y\_train)  
y\_pred\_rf = model\_rf.predict(X\_test)  
mse\_rf = mean\_squared\_error(y\_test, y\_pred\_rf)  
r2\_rf = r2\_score(y\_test, y\_pred\_rf)  
  
print("\nRandom Forest Results:")  
print(f"Mean Squared Error (Random Forest): {mse\_rf}")  
print(f"R-squared (Random Forest): {r2\_rf}")  
  
  
# Sample Predictions (First 5 Molecules in the Test Set)  
print("\nSample Predictions (First 5 Molecules in Test Set):")  
for i in range(min(5, len(y\_test))):  
 print(f"Molecule {i+1}: Actual pIC50 = {y\_test[i]:.2f}, Predicted pIC50 = {y\_pred[i]:.2f}")

**Explanation (English):**

* **Imports:** Imports necessary libraries (Pandas, RDKit, Scikit-learn).
* **File Path Handling:** Uses os.path.join to construct the correct path to your CSV file, ensuring it works regardless of your current working directory. It correctly uses os.path.abspath and os.getcwd() to define the base\_path.
* **Data Loading:** Loads the CSV data into a Pandas DataFrame, includes a try...except block to handle FileNotFoundError gracefully.
* **Data Cleaning:** Removes rows with missing canonical\_smiles or standard\_value. Converts ‘standard\_value’ to numeric, removing rows where conversion fails, ensures data is appropriate for processing.
* **RDKit Feature Generation:** Defines a function generate\_morgan\_fingerprint to create Morgan fingerprints (ECFP4) from SMILES strings. Handles potential errors during SMILES parsing. Applies the function to the DataFrame. Removes rows where fingerprint generation failed.
* **Model Training:**
  + Prepares the data for Scikit-learn by converting the fingerprints to a NumPy array and the standard\_value to a numpy array. Log transform the standard\_value.
  + Splits the data into training and testing sets using train\_test\_split.
  + Creates a LinearRegression model, trains it, and makes predictions on the test set.
  + Evaluates the model using Mean Squared Error (MSE) and R-squared.
  + An example using a RandomForestRegressor is also provided.
* **Sample Predictions:** Prints the actual and predicted activity values (pIC50, log-transformed IC50) for the first 5 molecules in the test set.
* **Error Handling**: Includes a try-except block for reading the CSV and a check for successful Morgan fingerprint generation.
* **Warnings Suppression**: Suppresses warnings to make the output cleaner.

**Explanation (Chinese):**

* **导入:** 导入必要的库（Pandas，RDKit，Scikit-learn）。
* **文件路径处理:** 使用 os.path.join 构建到 CSV 文件的正确路径，确保它与您当前的工作目录无关。 正确的使用了os.path.abspath 和 os.getcwd()来定义base\_path。
* **数据加载:** 将 CSV 数据加载到 Pandas DataFrame 中，包含一个 try...except 块来优雅地处理 FileNotFoundError。
* **数据清理:** 删除缺少 canonical\_smiles 或 standard\_value 的行。 将 ‘standard\_value’ 转换为数值，删除转换失败的行，确保数据适合处理。
* **RDKit 特征生成:** 定义一个函数 generate\_morgan\_fingerprint 以从 SMILES 字符串创建 Morgan 指纹（ECFP4）。 处理 SMILES 解析期间的潜在错误。 将该函数应用于 DataFrame。 删除指纹生成失败的行。
* **模型训练:**
  + 通过将指纹转换为 NumPy 数组，并将 standard\_value 转换为 numpy 数组，为 Scikit-learn 准备数据。 对 standard\_value 进行对数转换。
  + 使用 train\_test\_split 将数据拆分为训练集和测试集。
  + 创建一个 LinearRegression 模型，训练它，并在测试集上进行预测。
  + 使用均方误差 (MSE) 和 R 平方评估模型。
  + 还提供了一个使用 RandomForestRegressor 的示例。
* **样本预测:** 打印测试集中前 5 个分子的实际和预测的活性值（pIC50，对数转换的 IC50）。
* **错误处理**: 包括一个用于读取 CSV 的 try-except 块和一个用于检查 Morgan 指纹是否成功生成的块。
* **警告抑制**: 抑制警告以使输出更简洁。

**3. Addressing the Errors:**

* **ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard\_value ~ '^[0-9\.]+$':** **FIXED IN THE SQL CODE ABOVE.** The critical fix is to cast act.standard\_value to text before using the ~ operator (regular expression matching). PostgreSQL requires explicit casting when dealing with potentially mixed data types. This is done with act.standard\_value::text.
* **old scikit-learn version does not support parameters squared=False in the mean\_squared\_error function:** This is no longer relevant as the squared=False parameter is the default, but can also be removed from the mean\_squared\_error function call.

**4. Five Examples:**

Here are 5 examples of things you can do by modifying the code provided.

**Example 1: Different Activity Type**

Change the SQL query to extract data for a different activity type (e.g., Ki instead of IC50):

WHERE  
 act.standard\_type = 'Ki' -- Changed from IC50  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value::text ~ '^[0-9.]+$'  
LIMIT 100;

**Example 2: Different Molecular Descriptors**

Instead of Morgan fingerprints, use other RDKit descriptors:

from rdkit.Chem import Descriptors  
  
def calculate\_descriptors(smiles):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 descriptors = [Descriptors.MolLogP(mol), Descriptors.MolWt(mol), Descriptors.TPSA(mol)] #Example descriptors  
 return np.array(descriptors)  
 else:  
 return None  
 except:  
 return None  
  
df['descriptors'] = df['canonical\_smiles'].apply(calculate\_descriptors)  
df = df.dropna(subset=['descriptors'])  
  
X = np.vstack(df['descriptors'].to\_numpy())

**Example 3: Different Machine Learning Model**

Use a Support Vector Regression (SVR) model:

from sklearn.svm import SVR  
  
model = SVR(kernel='rbf') # Radial basis function kernel  
model.fit(X\_train, y\_train)  
y\_pred = model.predict(X\_test)

**Example 4: Cross-Validation**

Implement cross-validation for more robust model evaluation:

from sklearn.model\_selection import cross\_val\_score  
  
scores = cross\_val\_score(model, X, y, cv=5, scoring='neg\_mean\_squared\_error') #5-fold cross-validation  
  
print("Cross-validation scores (Negative MSE):", scores)  
print("Mean cross-validation score (Negative MSE):", scores.mean())

**Example 5: Feature Importance (for Random Forest)**

If you’re using a Random Forest model, extract and print feature importances:

if isinstance(model\_rf, RandomForestRegressor): #Check model type  
 importances = model\_rf.feature\_importances\_  
 print("Feature Importances:")  
 for i, importance in enumerate(importances):  
 print(f"Feature {i+1}: {importance:.4f}")

Remember to install necessary packages using pip install pandas rdkit scikit-learn.

This comprehensive guide should help you get your Chembl 35 data analysis project up and running smoothly. Remember to adapt the code to your specific research question and data. Good luck!