# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

Copyright 2025 | Nghiên Cứu Thuốc | www.nghiencuuthuoc.com | Zalo: +84888999311 |

## 🧩 Topic: Tìm target có hoạt tính đối kháng và đồng vận

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 này. Dựa trên thông tin bạn cung cấp, chúng ta sẽ tập trung vào chủ đề “Topic\_CheMBL\_35\_39” và giải quyết các vấn đề cụ thể bạn gặp phải.

**1. Phân tích mô hình phân tích (Analysis Model):**

Mục tiêu của chủ đề “Topic\_CheMBL\_35\_39” là gì? Để đưa ra mô hình phân tích phù hợp, chúng ta cần xác định rõ câu hỏi nghiên cứu. Dưới đây là một số khả năng:

* **Dự đoán hoạt tính (Activity Prediction):** Xây dựng mô hình dự đoán hoạt tính của các hợp chất dựa trên cấu trúc hóa học của chúng (ví dụ: dự đoán IC50, Ki, EC50).
* **Phân tích SAR (Structure-Activity Relationship):** Tìm hiểu mối quan hệ giữa cấu trúc hóa học của các hợp chất và hoạt tính sinh học của chúng. Xác định các nhóm chức (functional groups) hoặc phần tử cấu trúc (structural motifs) quan trọng ảnh hưởng đến hoạt tính.
* **Phân cụm hợp chất (Compound Clustering):** Phân nhóm các hợp chất thành các cụm dựa trên tính tương đồng về cấu trúc hoặc hoạt tính.
* **Phân tích QSAR/QSPR (Quantitative Structure-Activity/Property Relationship):** Xây dựng mô hình định lượng mối quan hệ giữa cấu trúc hóa học và hoạt tính/tính chất của hợp chất.

Dựa trên mục tiêu cụ thể, chúng ta có thể lựa chọn các phương pháp phân tích phù hợp:

* **Machine Learning:** Các thuật toán như Random Forest, Support Vector Machines (SVM), Neural Networks, Gradient Boosting.
* **Cheminformatics:** Sử dụng các descriptor phân tử (ví dụ: Morgan fingerprints, physicochemical properties) để biểu diễn cấu trúc hóa học và phân tích.
* **Statistical Analysis:** Các phương pháp thống kê như hồi quy tuyến tính, phân tích phương sai (ANOVA).

**2. Hướng dẫn song ngữ (Bilingual Guidance):**

**2.1. SQL:**

* **Purpose:** To extract relevant data from the ChEMBL database.
* **Mục đích:** Trích xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL.

**2.2. Python:**

* **Purpose:** To process the data, perform calculations, build models, and visualize results.
* **Mục đích:** Xử lý dữ liệu, thực hiện tính toán, xây dựng mô hình và trực quan hóa kết quả.

**3. Code SQL và Python mẫu (Sample SQL and Python Code):**

**3.1. SQL (with error handling):**

-- Extract data for Topic\_CheMBL\_35\_39 (limited to 100 records)  
-- Trích xuất dữ liệu cho Topic\_CheMBL\_35\_39 (giới hạn 100 bản ghi)  
  
SELECT  
 act.molregno,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units,  
 md.chembl\_id,  
 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 IN ('IC50', 'Ki', 'EC50') -- Example activity types  
 AND act.standard\_units = 'nM' -- Example units  
 AND act.standard\_value IS NOT NULL  
 AND cs.canonical\_smiles IS NOT NULL  
 -- Handle the numeric check safely:  
 AND act.standard\_value::text ~ '^[0-9\.]+$' -- Check if it's numeric  
LIMIT 100;

**Explanation:**

* The ::text cast is used to convert standard\_value to text before applying the regular expression.
* We limit the results to 100 records using LIMIT 100.

**3.2. Python (with scikit-learn version check):**

import pandas as pd  
import numpy as np  
from rdkit import Chem  
from rdkit.Chem import AllChem  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error  
import sklearn  
import os  
  
print(f"Scikit-learn version: {sklearn.\_\_version\_\_}")  
  
# Define paths (following AIMLOps Template)  
base\_path = "../data" # Adjust if needed  
csv\_file = os.path.join(base\_path, "chembl\_35\_data.csv") # Replace with your actual filename  
  
# Load data  
try:  
 data = pd.read\_csv(csv\_file)  
 print("Data loaded successfully.")  
except FileNotFoundError:  
 print(f"Error: File not found at {csv\_file}")  
 exit()  
  
# Data Cleaning and Preprocessing  
data = data.dropna(subset=['canonical\_smiles', 'standard\_value']) # Drop rows with missing SMILES or activity values  
data = data[data['standard\_value'].astype(str).str.match(r'^[0-9\.]+$')] #Keep numeric values only  
data['standard\_value'] = pd.to\_numeric(data['standard\_value']) # Transform to numeric  
  
# Function to convert SMILES to Morgan Fingerprints  
def smiles\_to\_morgan(smiles, radius=2, n\_bits=2048):  
 try:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol:  
 fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=n\_bits)  
 return np.array(list(fp.ToBitString()), dtype=int)  
 else:  
 return None  
 except:  
 return None  
  
# Apply the function to create fingerprints  
data['morgan\_fp'] = data['canonical\_smiles'].apply(smiles\_to\_morgan)  
data = data.dropna(subset=['morgan\_fp']) # Remove rows where fingerprint generation failed  
data = data[data['morgan\_fp'].map(len) > 0]  
  
# Prepare data for machine learning  
X = np.stack(data['morgan\_fp'].values)  
y = data['standard\_value'].values  
  
# 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 RandomForestRegressor model  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
# Make predictions  
y\_pred = model.predict(X\_test)  
  
# Calculate Mean Squared Error  
if sklearn.\_\_version\_\_ < '0.22':  
 mse = mean\_squared\_error(y\_test, y\_pred)  
else:  
 mse = mean\_squared\_error(y\_test, y\_pred, squared=False) # Use squared=False for newer versions  
  
print(f"Mean Squared Error: {mse}")

**Explanation:**

* **Error Handling:** Includes try...except blocks for file loading and SMILES conversion.
* **Scikit-learn Version Check:** Checks the scikit-learn version and uses the appropriate mean\_squared\_error call.
* **Data Preprocessing:** Handles missing data and converts SMILES to Morgan fingerprints.
* **Model Training:** Trains a Random Forest Regressor.
* **Evaluation:** Calculates and prints the Mean Squared Error.

**4. Ví dụ code SQL và Python mẫu (Sample SQL and Python Code Examples):**

Here are 5 examples of SQL and Python code snippets that you can adapt for your project.

**Example 1: Extracting compounds with specific properties**

* **SQL:** sql SELECT md.chembl\_id, cs.canonical\_smiles, act.standard\_value FROM molecule\_dictionary md JOIN compound\_structures cs ON md.molregno = cs.molregno JOIN activities act ON md.molregno = act.molregno WHERE md.mw\_freebase < 300 -- Molecular weight less than 300 AND act.standard\_type = 'IC50' LIMIT 10;
* **Python:** (Reading from a CSV file generated from the SQL query above) python import pandas as pd data = pd.read\_csv('../data/example1.csv') # Update with the correct path print(data.head())

**Example 2: Calculating LogP using RDKit**

* **SQL:** (Extract SMILES strings) sql SELECT md.chembl\_id, cs.canonical\_smiles FROM molecule\_dictionary md JOIN compound\_structures cs ON md.molregno = cs.molregno LIMIT 10;
* **Python:**
* import pandas as pd  
  from rdkit import Chem  
  from rdkit.Chem import Descriptors  
    
  data = pd.read\_csv('../data/example2.csv') # Update with the correct path  
    
  def calculate\_logp(smiles):  
   mol = Chem.MolFromSmiles(smiles)  
   if mol:  
   return Descriptors.MolLogP(mol)  
   else:  
   return None  
    
  data['logp'] = data['canonical\_smiles'].apply(calculate\_logp)  
  print(data.head())

**Example 3: Filtering by activity value range**

* **SQL:** sql SELECT md.chembl\_id, cs.canonical\_smiles, act.standard\_value 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' AND act.standard\_value BETWEEN 10 AND 100 -- IC50 between 10 and 100 nM LIMIT 10;
* **Python:** (Analyzing the extracted data)
* import pandas as pd  
  import matplotlib.pyplot as plt  
    
  data = pd.read\_csv('../data/example3.csv') # Update with the correct path  
  plt.hist(data['standard\_value'])  
  plt.xlabel('IC50 (nM)')  
  plt.ylabel('Frequency')  
  plt.title('IC50 Distribution')  
  plt.show()

**Example 4: Generating Morgan fingerprints and visualizing them**

* **SQL:** (Extract SMILES) sql SELECT md.chembl\_id, cs.canonical\_smiles FROM molecule\_dictionary md JOIN compound\_structures cs ON md.molregno = cs.molregno LIMIT 10;
* **Python:**
* import pandas as pd  
  from rdkit import Chem  
  from rdkit.Chem import AllChem  
  import matplotlib.pyplot as plt  
    
  data = pd.read\_csv('../data/example4.csv') # Update with the correct path  
    
  def generate\_morgan\_fp(smiles):  
   mol = Chem.MolFromSmiles(smiles)  
   if mol:  
   fp = AllChem.GetMorganFingerprint(mol, 2) # Radius 2  
   return fp  
   else:  
   return None  
    
  data['morgan\_fp'] = data['canonical\_smiles'].apply(generate\_morgan\_fp)  
  print(data.head())

**Example 5: Basic Activity Prediction with a Simple Model**

* **SQL:** sql SELECT md.chembl\_id, cs.canonical\_smiles, act.standard\_value 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' AND act.standard\_units = 'nM' LIMIT 100;
* **Python:** (Simple Activity Prediction)
* import pandas as pd  
  from rdkit import Chem  
  from rdkit.Chem import AllChem  
  from sklearn.model\_selection import train\_test\_split  
  from sklearn.linear\_model import LinearRegression  
  from sklearn.metrics import mean\_squared\_error  
  import numpy as np  
    
  # Load data  
  data = pd.read\_csv('../data/example5.csv') # Update with the correct path  
  data.dropna(inplace=True)  
    
  # RDKit: Generate Morgan Fingerprints  
  def generate\_fingerprint(smiles):  
   mol = Chem.MolFromSmiles(smiles)  
   if mol is not None:  
   fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius=2, nBits=2048)  
   return np.array(list(fp))  
   else:  
   return None  
    
  data['fingerprint'] = data['canonical\_smiles'].apply(generate\_fingerprint)  
  data.dropna(subset=['fingerprint'], inplace=True)  
    
  # Prepare data  
  X = np.stack(data['fingerprint'].values)  
  y = data['standard\_value'].values  
    
  # Split data  
  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
    
  # Model training  
  model = LinearRegression()  
  model.fit(X\_train, y\_train)  
    
  # Make predictions  
  y\_pred = model.predict(X\_test)  
    
  # Evaluate  
  mse = mean\_squared\_error(y\_test, y\_pred)  
  print(f'Mean Squared Error: {mse}')

**Important Considerations:**

* **Database Connection:** You’ll need to establish a connection to your PostgreSQL database using a Python library like psycopg2.
* **Data Scaling/Normalization:** Consider scaling or normalizing your activity values (e.g., converting IC50 to pIC50) to improve model performance.
* **Feature Selection:** Explore different molecular descriptors and feature selection techniques to optimize your models.
* **Model Validation:** Use appropriate validation techniques (e.g., cross-validation) to ensure the robustness of your models.
* **Logging:** Implement logging to track the progress of your analysis and debug any issues.
* **AIMLOps:** Integrate your code into the AIMLOps framework for reproducibility, version control, and automated deployment.

Hãy cho tôi biết nếu bạn có bất kỳ câu hỏi cụ thể nào hoặc muốn tôi đi sâu hơn vào một khía cạnh cụ thể của dự án. Chúc bạn thành công!