# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Tối ưu hóa XGBoost để dự đoán IC50

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Với vai trò là một chuyên gia R&D dược học, tôi sẽ cung cấp cho bạn hướng dẫn chi tiết, code mẫu và các phân tích cần thiết để bạn có thể triển khai dự án một cách hiệu quả.

**Tổng quan về dự án**

Dự án của bạn tập trung vào việc khai thác dữ liệu từ cơ sở dữ liệu ChEMBL 35, một nguồn tài nguyên vô giá cho việc nghiên cứu và phát triển thuốc. Bằng cách kết hợp ChEMBL với RDKit, một toolkit mã nguồn mở mạnh mẽ cho tin sinh học và hóa tin học, bạn có thể thực hiện các phân tích phức tạp về cấu trúc hóa học, hoạt tính sinh học và mối quan hệ cấu trúc-hoạt tính (SAR).

**Phân tích mô hình**

Mô hình phân tích của bạn sẽ bao gồm các bước sau:

1. **Trích xuất dữ liệu từ ChEMBL:** Sử dụng SQL để truy vấn và trích xuất dữ liệu liên quan đến các hợp chất và hoạt tính sinh học của chúng từ cơ sở dữ liệu ChEMBL 35.
2. **Tiền xử lý dữ liệu:** Làm sạch và chuẩn hóa dữ liệu, loại bỏ các giá trị thiếu hoặc không hợp lệ.
3. **Tính toán đặc trưng phân tử:** Sử dụng RDKit để tính toán các đặc trưng phân tử (molecular descriptors) từ cấu trúc hóa học của các hợp chất. Các đặc trưng này có thể bao gồm các thuộc tính vật lý hóa học, topo, hình học và điện tử.
4. **Phân tích SAR:** Áp dụng các phương pháp thống kê và học máy để xác định mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học. Ví dụ: bạn có thể sử dụng hồi quy tuyến tính, cây quyết định hoặc mạng nơ-ron để xây dựng các mô hình dự đoán hoạt tính dựa trên các đặc trưng phân tử.
5. **Trực quan hóa kết quả:** Sử dụng các công cụ trực quan hóa dữ liệu để trình bày kết quả phân tích một cách rõ ràng và dễ hiểu. Ví dụ: bạn có thể tạo các biểu đồ phân tán, biểu đồ hộp hoặc bản đồ nhiệt để khám phá các xu hướng và mối quan hệ trong dữ liệu.

**Hướng dẫn song ngữ và code mẫu**

Dưới đây là các ví dụ về code SQL và Python, kèm theo giải thích chi tiết bằng cả tiếng Anh và tiếng Việt:

**1. Trích xuất dữ liệu từ ChEMBL (SQL)**

-- SQL query to extract data from ChEMBL database  
-- Truy vấn SQL để trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_type,  
 act.standard\_value,  
 act.standard\_units  
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'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value ~ '^[0-9\.]+$' -- Lọc giá trị số  
LIMIT 100;

**Giải thích:**

* **SELECT:** Chọn các cột molregno, canonical\_smiles, standard\_type, standard\_value và standard\_units từ các bảng molecule\_dictionary, compound\_structures và activities.
* **FROM:** Chỉ định các bảng để truy vấn.
* **JOIN:** Kết hợp các bảng dựa trên khóa ngoại molregno.
* **WHERE:** Lọc dữ liệu dựa trên các điều kiện sau:
  + act.standard\_type = 'IC50': Chỉ chọn các hoạt tính có loại là IC50.
  + act.standard\_units = 'nM': Chỉ chọn các hoạt tính có đơn vị là nM.
  + act.standard\_value IS NOT NULL: Loại bỏ các giá trị standard\_value bị thiếu.
  + act.standard\_value ~ '^[0-9\.]+$': Chỉ chọn các giá trị standard\_value là số.
* **LIMIT 100:** Giới hạn số lượng kết quả trả về là 100 dòng.

**Explanation:**

* **SELECT:** Selects the molregno, canonical\_smiles, standard\_type, standard\_value, and standard\_units columns from the molecule\_dictionary, compound\_structures, and activities tables.
* **FROM:** Specifies the tables to query.
* **JOIN:** Joins the tables based on the foreign key molregno.
* **WHERE:** Filters the data based on the following conditions:
  + act.standard\_type = 'IC50': Only selects activities with type IC50.
  + act.standard\_units = 'nM': Only selects activities with units nM.
  + act.standard\_value IS NOT NULL: Removes missing standard\_value values.
  + act.standard\_value ~ '^[0-9\.]+$': Only selects standard\_value values that are numeric.
* **LIMIT 100:** Limits the number of results returned to 100 rows.

**Sửa lỗi:**

Lỗi ERROR: operator does not exist: numeric ~ unknown xảy ra do bạn đang cố gắng so sánh một cột kiểu số (numeric) với một chuỗi (unknown). Để khắc phục, bạn có thể ép kiểu cột standard\_value sang kiểu văn bản trước khi thực hiện so sánh:

AND CAST(act.standard\_value AS TEXT) ~ '^[0-9\.]+$'

**2. Tiền xử lý dữ liệu và tính toán đặc trưng phân tử (Python)**

# Python code to preprocess data and calculate molecular descriptors using RDKit  
# Mã Python để tiền xử lý dữ liệu và tính toán các đặc trưng phân tử bằng RDKit  
import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
import numpy as np  
  
# Define the base path  
# Xác định đường dẫn gốc  
base\_path = "../data"  
  
# Read the data from the CSV file  
# Đọc dữ liệu từ file CSV  
csv\_file = os.path.join(base\_path, "chembl\_ic50\_data.csv") # Thay đổi tên file cho phù hợp  
data = pd.read\_csv(csv\_file)  
  
# Function to calculate molecular descriptors  
# Hàm tính toán các đặc trưng phân tử  
def calculate\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 descriptors = {}  
 descriptors['MolecularWeight'] = Descriptors.MolWt(mol)  
 descriptors['LogP'] = Descriptors.MolLogP(mol)  
 descriptors['HBD'] = Descriptors.NumHDonors(mol)  
 descriptors['HBA'] = Descriptors.NumHAcceptors(mol)  
 return descriptors  
 else:  
 return None  
  
# Apply the function to each SMILES string  
# Áp dụng hàm cho mỗi chuỗi SMILES  
data['descriptors'] = data['canonical\_smiles'].apply(calculate\_descriptors)  
  
# Convert descriptors to columns  
# Chuyển đổi các đặc trưng thành các cột  
data = pd.concat([data.drop(['descriptors'], axis=1), data['descriptors'].apply(pd.Series)], axis=1)  
  
# Clean up data (remove rows with missing descriptors)  
# Làm sạch dữ liệu (loại bỏ các hàng có đặc trưng bị thiếu)  
data = data.dropna(subset=['MolecularWeight', 'LogP', 'HBD', 'HBA'])  
  
# Display the first few rows of the processed data  
# Hiển thị một vài hàng đầu tiên của dữ liệu đã được xử lý  
print(data.head())

**Giải thích:**

* **Import libraries:** Nhập các thư viện cần thiết, bao gồm pandas để thao tác dữ liệu, rdkit để tính toán đặc trưng phân tử và numpy cho các phép toán số học.
* **Read data:** Đọc dữ liệu từ file CSV đã tạo ở bước trước bằng SQL.
* **calculate\_descriptors function:** Hàm này nhận một chuỗi SMILES làm đầu vào, tạo một đối tượng phân tử từ chuỗi SMILES bằng RDKit, và tính toán các đặc trưng phân tử như trọng lượng phân tử (MolecularWeight), hệ số phân vùng octanol-nước (LogP), số lượng liên kết hydro cho (HBD) và nhận (HBA).
* **Apply function:** Áp dụng hàm calculate\_descriptors cho mỗi chuỗi SMILES trong cột canonical\_smiles của DataFrame.
* **Convert descriptors to columns:** Chuyển đổi các đặc trưng phân tử thành các cột riêng biệt trong DataFrame.
* **Clean up data:** Loại bỏ các hàng có giá trị đặc trưng bị thiếu.
* **Display data:** Hiển thị một vài hàng đầu tiên của dữ liệu đã được xử lý.

**Explanation:**

* **Import libraries:** Imports the necessary libraries, including pandas for data manipulation, rdkit for calculating molecular descriptors, and numpy for numerical operations.
* **Read data:** Reads the data from the CSV file created in the previous step using SQL.
* **calculate\_descriptors function:** This function takes a SMILES string as input, creates a molecule object from the SMILES string using RDKit, and calculates molecular descriptors such as molecular weight (MolecularWeight), octanol-water partition coefficient (LogP), number of hydrogen bond donors (HBD), and number of hydrogen bond acceptors (HBA).
* **Apply function:** Applies the calculate\_descriptors function to each SMILES string in the canonical\_smiles column of the DataFrame.
* **Convert descriptors to columns:** Converts the molecular descriptors into separate columns in the DataFrame.
* **Clean up data:** Removes rows with missing descriptor values.
* **Display data:** Displays the first few rows of the processed data.

**Sửa lỗi:**

Lỗi TypeError: mean\_squared\_error() got an unexpected keyword argument 'squared' xảy ra do phiên bản scikit-learn bạn đang sử dụng quá cũ và không hỗ trợ tham số squared. Để khắc phục, bạn có thể bỏ tham số squared hoặc nâng cấp phiên bản scikit-learn của mình.

**3. Phân tích SAR (Python)**

# Python code for SAR analysis using scikit-learn  
# Mã Python để phân tích SAR bằng scikit-learn  
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  
  
# Prepare the data  
# Chuẩn bị dữ liệu  
X = data[['MolecularWeight', 'LogP', 'HBD', 'HBA']]  
y = data['standard\_value']  
  
# Scale the features  
# Chuẩn hóa các đặc trưng  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# Split the data into training and testing sets  
# 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)  
  
# Create a linear regression model  
# Tạo một mô hình hồi quy tuyến tính  
model = LinearRegression()  
  
# Train the model  
# Huấn luyện mô hình  
model.fit(X\_train, y\_train)  
  
# Make predictions on the test set  
# Dự đoán trên tập kiểm tra  
y\_pred = model.predict(X\_test)  
  
# Evaluate the model  
# 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}")

**Giải thích:**

* **Import libraries:** Nhập các thư viện cần thiết từ scikit-learn, bao gồm train\_test\_split để chia dữ liệu, LinearRegression để xây dựng mô hình hồi quy tuyến tính, mean\_squared\_error và r2\_score để đánh giá mô hình, và StandardScaler để chuẩn hóa dữ liệu.
* **Prepare data:** Chuẩn bị dữ liệu bằng cách chọn các đặc trưng phân tử (MolecularWeight, LogP, HBD, HBA) làm biến độc lập (X) và giá trị hoạt tính (standard\_value) làm biến phụ thuộc (y).
* **Scale features:** Chuẩn hóa các đặc trưng bằng StandardScaler để đảm bảo rằng tất cả các đặc trưng đều có cùng tỷ lệ.
* **Split data:** Chia dữ liệu thành tập huấn luyện (80%) và tập kiểm tra (20%).
* **Create model:** Tạo một mô hình hồi quy tuyến tính.
* **Train model:** Huấn luyện mô hình trên tập huấn luyện.
* **Make predictions:** Dự đoán giá trị hoạt tính trên tập kiểm tra.
* **Evaluate model:** Đánh giá mô hình bằng cách tính toán sai số bình phương trung bình (mse) và hệ số xác định (r2).

**Explanation:**

* **Import libraries:** Imports the necessary libraries from scikit-learn, including train\_test\_split for splitting the data, LinearRegression for building a linear regression model, mean\_squared\_error and r2\_score for evaluating the model, and StandardScaler for scaling the data.
* **Prepare data:** Prepares the data by selecting the molecular descriptors (MolecularWeight, LogP, HBD, HBA) as independent variables (X) and the activity value (standard\_value) as the dependent variable (y).
* **Scale features:** Scales the features using StandardScaler to ensure that all features are on the same scale.
* **Split data:** Splits the data into training (80%) and testing (20%) sets.
* **Create model:** Creates a linear regression model.
* **Train model:** Trains the model on the training set.
* **Make predictions:** Predicts the activity values on the test set.
* **Evaluate model:** Evaluates the model by calculating the mean squared error (mse) and the R-squared coefficient (r2).

**4. Trực quan hóa kết quả (Python)**

# Python code to visualize the results  
# Mã Python để trực quan hóa kết quả  
import matplotlib.pyplot as plt  
  
# Create a scatter plot of predicted vs. actual values  
# Tạo biểu đồ phân tán giữa giá trị dự đoán và giá trị thực tế  
plt.scatter(y\_test, y\_pred)  
plt.xlabel("Actual IC50 (nM)")  
plt.ylabel("Predicted IC50 (nM)")  
plt.title("Actual vs. Predicted IC50 Values")  
plt.show()

**5. Ví dụ code SQL và Python mẫu**

Dưới đây là 5 ví dụ code SQL và Python mẫu để bạn tham khảo:

**Ví dụ 1: Trích xuất dữ liệu về các hợp chất có hoạt tính chống lại một mục tiêu cụ thể (SQL)**

-- SQL query to extract data for compounds active against a specific target  
-- Truy vấn SQL để trích xuất dữ liệu cho các hợp chất có hoạt tính chống lại một mục tiêu cụ thể  
SELECT  
 md.molregno,  
 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'  
 AND act.target\_chembl\_id = 'CHEMBL205' -- Replace with the desired target ChEMBL ID  
LIMIT 100;

**Ví dụ 2: Tính toán số lượng vòng trong phân tử (Python)**

# Python code to calculate the number of rings in a molecule  
# Mã Python để tính toán số lượng vòng trong phân tử  
from rdkit import Chem  
from rdkit.Chem import rdMolDescriptors  
  
def calculate\_num\_rings(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 return rdMolDescriptors.CalcNumRings(mol)  
 else:  
 return None  
  
# Example usage  
# Ví dụ sử dụng  
smiles = 'CC(=O)Oc1ccccc1C(=O)O'  
num\_rings = calculate\_num\_rings(smiles)  
print(f"Number of rings in {smiles}: {num\_rings}")

**Ví dụ 3: Tìm các hợp chất có trọng lượng phân tử nằm trong một khoảng nhất định (SQL)**

-- SQL query to find compounds with molecular weight within a certain range  
-- Truy vấn SQL để tìm các hợp chất có trọng lượng phân tử nằm trong một khoảng nhất định  
SELECT  
 md.molregno,  
 cs.canonical\_smiles  
FROM  
 molecule\_dictionary md  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
WHERE  
 md.mw\_freebase BETWEEN 200 AND 400  
LIMIT 100;

**Ví dụ 4: Tính toán các đặc trưng phân tử bổ sung (Python)**

# Python code to calculate additional molecular descriptors  
# Mã Python để tính toán các đặc trưng phân tử bổ sung  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def calculate\_additional\_descriptors(smiles):  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is not None:  
 descriptors = {}  
 descriptors['TPSA'] = Descriptors.TPSA(mol) # Diện tích bề mặt phân cực topo  
 descriptors['QED'] = Descriptors.qed(mol) # Chất lượng giống thuốc  
 return descriptors  
 else:  
 return None

**Ví dụ 5: Xây dựng mô hình dự đoán hoạt tính bằng Random Forest (Python)**

# Python code to build a Random Forest model for activity prediction  
# Mã Python để xây dựng mô hình Random Forest để dự đoán hoạt tính  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.preprocessing import StandardScaler  
import pandas as pd  
  
# Assuming 'data' DataFrame is already loaded and preprocessed  
# Giả sử DataFrame 'data' đã được tải và tiền xử lý  
  
# Prepare the data  
# Chuẩn bị dữ liệu  
X = data[['MolecularWeight', 'LogP', 'HBD', 'HBA']]  
y = data['standard\_value']  
  
# Scale the features  
# Chuẩn hóa các đặc trưng  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# Split the data into training and testing sets  
# 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)  
  
# Create a Random Forest model  
# Tạo một mô hình Random Forest  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
  
# Train the model  
# Huấn luyện mô hình  
model.fit(X\_train, y\_train)  
  
# Make predictions on the test set  
# Dự đoán trên tập kiểm tra  
y\_pred = model.predict(X\_test)  
  
# Evaluate the model  
# 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}")

**Lưu ý quan trọng:**

* Hãy nhớ thay đổi tên file CSV trong code Python cho phù hợp với tên file bạn đã lưu.
* Bạn có thể tùy chỉnh các đặc trưng phân tử được tính toán và các tham số của mô hình học máy để phù hợp với mục tiêu nghiên cứu của mình.
* Hãy sử dụng các phương pháp đánh giá mô hình phù hợp để đảm bảo rằng mô hình của bạn có khả năng dự đoán tốt trên dữ liệu mới.

Chúc bạn thành công với dự án của mình! Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi.