# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

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

## 🧩 Topic: Sàng lọc ảo dựa trên các target trong ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 này. Với vai trò là chuyên gia nghiên cứu và phát triển dược học, tôi sẽ tập trung vào việc cung cấp các phân tích, hướng dẫn và mã nguồn (SQL và Python) chi tiết, dễ hiểu để bạn có thể tận dụng tối đa dữ liệu ChEMBL 35 cho nghiên cứu phát triển thuốc của mình.

**Tổng quan dự án và mục tiêu:**

Dựa trên thông tin bạn cung cấp, chúng ta sẽ thực hiện các bước sau:

1. **Kết nối và truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35 (PostgreSQL):** Sử dụng thông tin đăng nhập bạn cung cấp để truy cập dữ liệu cần thiết.
2. **Tiền xử lý dữ liệu:** Làm sạch và chuẩn hóa dữ liệu để đảm bảo chất lượng cho các phân tích tiếp theo.
3. **Phân tích dữ liệu:** Sử dụng RDKit để tính toán các descriptor phân tử và kết hợp với dữ liệu hoạt tính sinh học từ ChEMBL để xây dựng mô hình dự đoán.
4. **Lưu trữ và quản lý dữ liệu:** Sử dụng định dạng CSV để lưu trữ dữ liệu đã xử lý, tuân thủ cấu trúc thư mục theo chuẩn AIMLOps.
5. **Xây dựng mô hình:** Sử dụng scikit-learn để xây dựng mô hình dự đoán hoạt tính.
6. **Đánh giá mô hình:** Sử dụng các metrics phù hợp để đánh giá hiệu năng của mô hình.

**Phân tích chủ đề Topic\_CheMBL\_35\_6**

Chủ đề này không được định nghĩa rõ ràng. Để đưa ra phân tích cụ thể, tôi cần biết chủ đề Topic\_CheMBL\_35\_6 tập trung vào vấn đề gì (ví dụ: dự đoán hoạt tính của một target cụ thể, phân tích mối quan hệ cấu trúc-hoạt tính (SAR), v.v.). Tuy nhiên, tôi sẽ cung cấp một ví dụ tổng quát về quy trình phân tích dữ liệu ChEMBL để bạn tham khảo.

**Ví dụ tổng quát: Dự đoán hoạt tính ức chế của hợp chất đối với một target (ví dụ: Kinase X)**

**1. Mô hình phân tích:**

* **Mục tiêu:** Xây dựng mô hình dự đoán khả năng ức chế Kinase X của một hợp chất dựa trên cấu trúc phân tử của nó.
* **Dữ liệu:**
  + Dữ liệu hoạt tính sinh học (ví dụ: IC50) của các hợp chất đối với Kinase X từ cơ sở dữ liệu ChEMBL.
  + Cấu trúc phân tử của các hợp chất (SMILES).
* **Phương pháp:**
  + **Tính toán descriptor phân tử:** Sử dụng RDKit để tính toán các descriptor (ví dụ: MW, LogP, HBD, HBA) từ cấu trúc SMILES.
  + **Xây dựng mô hình:** Sử dụng thuật toán học máy (ví dụ: Random Forest, Support Vector Machine) để xây dựng mô hình dự đoán IC50 dựa trên các descriptor đã tính toán.
* **Đánh giá mô hình:** Sử dụng các metrics như RMSE, R-squared, AUC để đánh giá hiệu năng của mô hình trên tập kiểm tra.

**2. Hướng dẫn song ngữ (Vietnamese - English):**

* **Kết nối đến cơ sở dữ liệu ChEMBL (Connect to ChEMBL database):**
  + *Tiếng Việt:* Sử dụng thư viện psycopg2 để kết nối đến cơ sở dữ liệu PostgreSQL ChEMBL 35.
  + *English:* Use the psycopg2 library to connect to the ChEMBL 35 PostgreSQL database.
* **Truy vấn dữ liệu hoạt tính và cấu trúc (Query activity and structure data):**
  + *Tiếng Việt:* Viết câu lệnh SQL để truy vấn dữ liệu hoạt tính (ví dụ: IC50) của các hợp chất đối với Kinase X và cấu trúc SMILES của chúng.
  + *English:* Write an SQL query to retrieve activity data (e.g., IC50) of compounds against Kinase X and their SMILES structures.
* **Tính toán descriptor phân tử (Calculate molecular descriptors):**
  + *Tiếng Việt:* Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc SMILES.
  + *English:* Use RDKit to calculate molecular descriptors from SMILES structures.
* **Xây dựng mô hình học máy (Build machine learning model):**
  + *Tiếng Việt:* Sử dụng scikit-learn để xây dựng mô hình dự đoán IC50 dựa trên các descriptor đã tính toán.
  + *English:* Use scikit-learn to build a model to predict IC50 based on the calculated descriptors.
* **Đánh giá mô hình (Evaluate the model):**
  + *Tiếng Việt:* Sử dụng các metrics như RMSE, R-squared để đánh giá hiệu năng của mô hình.
  + *English:* Use metrics like RMSE, R-squared to evaluate the model’s performance.

**3. Mã nguồn (SQL & Python):**

**SQL (Lấy 100 dòng dữ liệu):**

-- English: Query to retrieve 100 activity data points and their corresponding SMILES for Kinase X  
-- Vietnamese: Truy vấn để lấy 100 dữ liệu hoạt tính và cấu trúc SMILES tương ứng cho Kinase X  
  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 activities act  
JOIN  
 molecule\_dictionary md ON act.molregno = md.molregno  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Kinase X' -- Thay 'Kinase X' bằng tên target thực tế  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0  
 AND act.standard\_relation = '='  
 AND act.standard\_value ~ '^[0-9\.]+$' --Lọc giá trị số (Khắc phục lỗi a)  
LIMIT 100;

**Lưu ý về lỗi SQL:**

Lỗi ERROR: operator does not exist: numeric ~ unknown xảy ra do PostgreSQL không thể so sánh trực tiếp kiểu dữ liệu numeric với một biểu thức regular expression (kiểu unknown). Để khắc phục, bạn có thể sử dụng hàm CAST để chuyển đổi cột standard\_value sang kiểu TEXT trước khi so sánh với regular expression, hoặc sử dụng SIMILAR TO thay vì ~. Tuy nhiên, cách tốt nhất là lọc trước để đảm bảo cột standard\_value chỉ chứa giá trị số. Câu lệnh trên đã được sửa để đảm bảo rằng chỉ các giá trị số mới được chọn.

**Python (Jupyter Notebook):**

# -\*- coding: utf-8 -\*-  
import os  
import psycopg2  
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.ensemble import RandomForestRegressor  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.preprocessing import StandardScaler  
  
# Cấu hình kết nối database (Database connection configuration)  
db\_params = {  
 'host': '192.168.206.136',  
 'user': 'rd',  
 'password': 'rd',  
 'database': 'chembl\_35'  
}  
  
# Đường dẫn cơ sở (Base path)  
base\_path = '../data'  
  
# Tên file CSV (CSV filename)  
csv\_filename = 'kinase\_x\_data.csv'  
csv\_filepath = os.path.join(base\_path, csv\_filename)  
  
def fetch\_data\_from\_chembl(db\_params, query, csv\_filepath):  
 """  
 Fetches data from ChEMBL database using the provided SQL query and saves it to a CSV file.  
 Args:  
 db\_params (dict): Dictionary containing database connection parameters.  
 query (str): SQL query to execute.  
 csv\_filepath (str): Path to save the fetched data as CSV.  
 Returns:  
 pandas.DataFrame: DataFrame containing the fetched data.  
 """  
 try:  
 conn = psycopg2.connect(\*\*db\_params)  
 df = pd.read\_sql\_query(query, conn)  
 df.to\_csv(csv\_filepath, index=False)  
 conn.close()  
 print(f"Data saved to {csv\_filepath}")  
 return df  
 except psycopg2.Error as e:  
 print(f"Error connecting to database: {e}")  
 return None  
  
  
def calculate\_descriptors(smiles):  
 """Calculates molecular descriptors using RDKit.  
  
 Args:  
 smiles (str): SMILES string of the molecule.  
  
 Returns:  
 dict: A dictionary of calculated descriptors. Returns None if the molecule is invalid.  
 """  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
  
 descriptors = {}  
 descriptors["MolLogP"] = Descriptors.MolLogP(mol)  
 descriptors["MolWt"] = Descriptors.MolWt(mol)  
 descriptors["NumHAcceptors"] = Descriptors.NumHAcceptors(mol)  
 descriptors["NumHDonors"] = Descriptors.NumHDonors(mol)  
 descriptors["TPSA"] = Descriptors.TPSA(mol)  
 return descriptors  
  
def prepare\_data(df):  
 """Prepares the data for machine learning.  
  
 Args:  
 df (pandas.DataFrame): DataFrame containing SMILES and activity data.  
  
 Returns:  
 pandas.DataFrame: DataFrame containing molecular descriptors and activity values.  
 """  
 # Apply descriptor calculation and handle errors  
 descriptors\_list = []  
 for smiles in df['canonical\_smiles']:  
 descriptors = calculate\_descriptors(smiles)  
 if descriptors is not None:  
 descriptors\_list.append(descriptors)  
 else:  
 descriptors\_list.append(None) # Append None for invalid SMILES  
  
 # Create a DataFrame from the descriptor list  
 descriptors\_df = pd.DataFrame(descriptors\_list)  
  
 # Concatenate descriptors with the original DataFrame  
 df = pd.concat([df, descriptors\_df], axis=1)  
  
 # Drop rows with NaN values in descriptors (invalid SMILES)  
 df = df.dropna(subset=descriptors\_df.columns)  
  
 # Convert standard\_value to numeric, handling errors  
 df['standard\_value'] = pd.to\_numeric(df['standard\_value'], errors='coerce')  
 df = df.dropna(subset=['standard\_value']) # Drop rows where conversion failed  
  
 return df  
  
def train\_and\_evaluate\_model(df):  
 """Trains and evaluates a Random Forest Regressor model.  
  
 Args:  
 df (pandas.DataFrame): DataFrame containing descriptors and activity values.  
 """  
 # Prepare features (X) and target (y)  
 X = df[['MolLogP', 'MolWt', 'NumHAcceptors', 'NumHDonors', 'TPSA']]  
 y = np.log10(df['standard\_value']) # Convert IC50 to pIC50  
  
 # 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)  
  
 # Feature scaling  
 scaler = StandardScaler()  
 X\_train = scaler.fit\_transform(X\_train)  
 X\_test = scaler.transform(X\_test)  
  
 # Train the Random Forest Regressor model  
 model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
 model.fit(X\_train, y\_train)  
  
 # Make predictions  
 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}")  
  
  
# SQL query to fetch data (limit to 100 rows)  
query = """  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 activities act  
JOIN  
 molecule\_dictionary md ON act.molregno = md.molregno  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Kinase X' -- Replace 'Kinase X' with the actual target name  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0  
 AND act.standard\_relation = '='  
 AND act.standard\_value ~ '^[0-9\.]+$'  
LIMIT 100;  
"""  
  
# Lấy dữ liệu từ Chembl và lưu vào file CSV  
df = fetch\_data\_from\_chembl(db\_params, query, csv\_filepath)  
  
# Chuẩn bị dữ liệu  
if df is not None:  
 df = prepare\_data(df)  
  
 # Huấn luyện và đánh giá mô hình  
 train\_and\_evaluate\_model(df)

**Lưu ý về lỗi Python:**

1. **Phiên bản scikit-learn cũ:** Nếu bạn gặp lỗi squared=False trong mean\_squared\_error, hãy cập nhật scikit-learn lên phiên bản mới nhất bằng lệnh pip install -U scikit-learn. Nếu không thể cập nhật, hãy bỏ tham số squared=False (mặc định là True, trả về MSE thay vì RMSE). Trong code trên, tôi đã bỏ tham số này để tương thích với các phiên bản scikit-learn cũ.
2. **Xử lý lỗi SMILES:** Code trên đã bao gồm phần xử lý lỗi khi tính toán descriptor từ SMILES. Nếu một SMILES không hợp lệ, chương trình sẽ bỏ qua dòng đó.

**4. Ví dụ mã nguồn (SQL & Python) mẫu:**

Dưới đây là 5 ví dụ về các truy vấn SQL và mã Python khác nhau bạn có thể sử dụng:

**Ví dụ 1: Lấy dữ liệu hoạt tính cho một loạt các targets**

* **SQL:**

-- English: Get activity data for a list of targets  
-- Vietnamese: Lấy dữ liệu hoạt tính cho một danh sách các targets  
  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units,  
 td.pref\_name AS target\_name  
FROM  
 activities act  
JOIN  
 molecule\_dictionary md ON act.molregno = md.molregno  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name IN ('Kinase X', 'Enzyme Y', 'Receptor Z') -- Thay đổi tên targets tùy ý  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value IS NOT NULL  
 AND act.standard\_value > 0  
 AND act.standard\_relation = '='  
LIMIT 100;

* **Python:** (Sử dụng kết quả truy vấn trên để phân tích SAR đơn giản)

import pandas as pd  
from rdkit import Chem  
from rdkit.Chem import Draw  
  
def visualize\_compounds\_by\_activity(df, target\_name, num\_compounds=4):  
 """Visualizes compounds with high and low activity for a given target.  
  
 Args:  
 df (pandas.DataFrame): DataFrame containing SMILES, activity, and target name.  
 target\_name (str): Name of the target to visualize.  
 num\_compounds (int): Number of compounds to visualize for each activity group.  
 """  
 target\_df = df[df['target\_name'] == target\_name].copy()  
 if target\_df.empty:  
 print(f"No data found for target: {target\_name}")  
 return  
  
 # Convert IC50 to numeric, handling errors  
 target\_df['standard\_value'] = pd.to\_numeric(target\_df['standard\_value'], errors='coerce')  
 target\_df.dropna(subset=['standard\_value'], inplace=True)  
  
 # Sort by activity (IC50)  
 target\_df = target\_df.sort\_values(by='standard\_value')  
  
 # Get top and bottom compounds  
 top\_compounds = target\_df.head(num\_compounds)  
 bottom\_compounds = target\_df.tail(num\_compounds)  
  
 # Create molecule objects  
 top\_mols = [Chem.MolFromSmiles(s) for s in top\_compounds['canonical\_smiles']]  
 bottom\_mols = [Chem.MolFromSmiles(s) for s in bottom\_compounds['canonical\_smiles']]  
  
 # Draw molecules  
 img\_top = Draw.MolsToGridImage(top\_mols, molsPerRow=num\_compounds,  
 legends=[f"IC50: {v:.2f} nM" for v in top\_compounds['standard\_value']],  
 subImgSize=(200, 200),  
 title=f"Top {num\_compounds} most active compounds for {target\_name}")  
  
 img\_bottom = Draw.MolsToGridImage(bottom\_mols, molsPerRow=num\_compounds,  
 legends=[f"IC50: {v:.2f} nM" for v in bottom\_compounds['standard\_value']],  
 subImgSize=(200, 200),  
 title=f"Top {num\_compounds} least active compounds for {target\_name}")  
  
 return img\_top, img\_bottom  
  
# Assuming you have a DataFrame 'df' with the SQL query results  
# Example usage:  
# img\_top, img\_bottom = visualize\_compounds\_by\_activity(df, 'Kinase X')  
# img\_top # Display the top compounds  
# img\_bottom # Display the bottom compounds

**Ví dụ 2: Tính toán và phân tích phân bố descriptor**

* **SQL:** (Sử dụng lại truy vấn ở trên để lấy SMILES)
* **Python:**

import pandas as pd  
import matplotlib.pyplot as plt  
from rdkit import Chem  
from rdkit.Chem import Descriptors  
  
def calculate\_descriptors(smiles):  
 """Calculates molecular descriptors using RDKit."""  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 return None  
  
 descriptors = {}  
 descriptors["MolLogP"] = Descriptors.MolLogP(mol)  
 descriptors["MolWt"] = Descriptors.MolWt(mol)  
 return descriptors  
  
def analyze\_descriptor\_distribution(df):  
 """Calculates descriptors and plots their distribution.  
  
 Args:  
 df (pandas.DataFrame): DataFrame containing SMILES.  
 """  
 descriptors = []  
 for smiles in df['canonical\_smiles']:  
 desc = calculate\_descriptors(smiles)  
 if desc:  
 descriptors.append(desc)  
  
 descriptors\_df = pd.DataFrame(descriptors)  
 descriptors\_df.hist()  
 plt.show()  
  
# Example usage:  
# analyze\_descriptor\_distribution(df)

**Ví dụ 3: Lọc dữ liệu theo khoảng giá trị hoạt tính**

* **SQL:**

-- English: Filter data by activity range  
-- Vietnamese: Lọc dữ liệu theo khoảng giá trị hoạt tính  
  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 activities act  
JOIN  
 molecule\_dictionary md ON act.molregno = md.molregno  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Kinase X'  
 AND act.standard\_type = 'IC50'  
 AND act.standard\_units = 'nM'  
 AND act.standard\_value BETWEEN 10 AND 100 -- Lọc IC50 từ 10 nM đến 100 nM  
 AND act.standard\_relation = '='  
LIMIT 100;

* **Python:** (Không cần, vì đã lọc bằng SQL)

**Ví dụ 4: Tìm các hợp chất có khung cấu trúc tương tự**

* **SQL:** (Không thể thực hiện trực tiếp, cần Python để tạo fingerprints)
* **Python:**

from rdkit import Chem  
from rdkit.Chem import AllChem  
from rdkit.DataStructs import FingerprintSimilarity  
import pandas as pd  
  
def find\_similar\_compounds(df, smiles\_query, num\_results=5):  
 """Finds compounds in the DataFrame that are structurally similar to a query compound.  
  
 Args:  
 df (pandas.DataFrame): DataFrame containing SMILES.  
 smiles\_query (str): SMILES string of the query compound.  
 num\_results (int): Number of similar compounds to return.  
  
 Returns:  
 pandas.DataFrame: DataFrame containing the most similar compounds.  
 """  
 mol\_query = Chem.MolFromSmiles(smiles\_query)  
 if mol\_query is None:  
 print("Invalid SMILES for query compound")  
 return None  
  
 fp\_query = AllChem.GetMorganFingerprint(mol\_query, radius=2) # Morgan fingerprint  
  
 similarities = []  
 for smiles in df['canonical\_smiles']:  
 mol = Chem.MolFromSmiles(smiles)  
 if mol is None:  
 similarities.append(0.0) # Assign similarity of 0 to invalid SMILES  
 continue  
 fp = AllChem.GetMorganFingerprint(mol, radius=2)  
 similarity = FingerprintSimilarity(fp\_query, fp)  
 similarities.append(similarity)  
  
 df['similarity'] = similarities  
 df\_sorted = df.sort\_values(by='similarity', ascending=False)  
 return df\_sorted.head(num\_results)  
  
# Example Usage:  
# query\_smiles = "CCOc1ccccc1C(=O)Oc2ccccc2C(=O)O" # Example SMILES  
# similar\_compounds = find\_similar\_compounds(df, query\_smiles)  
# print(similar\_compounds)

**Ví dụ 5: Kết hợp dữ liệu ChEMBL với dữ liệu bên ngoài (ví dụ: dữ liệu độc tính)**

* **SQL:** (Lấy dữ liệu ChEMBL)

-- English: Get activity data for specific compounds  
-- Vietnamese: Lấy dữ liệu hoạt tính cho các hợp chất cụ thể  
  
SELECT  
 md.molregno,  
 cs.canonical\_smiles,  
 act.standard\_value,  
 act.standard\_units  
FROM  
 activities act  
JOIN  
 molecule\_dictionary md ON act.molregno = md.molregno  
JOIN  
 compound\_structures cs ON md.molregno = cs.molregno  
JOIN  
 target\_dictionary td ON act.tid = td.tid  
WHERE  
 td.pref\_name = 'Kinase X'  
 AND act.standard\_type = 'IC50'  
 AND md.chembl\_id IN ('CHEMBL1', 'CHEMBL2', 'CHEMBL3') -- Thay đổi chembl\_id tùy ý  
LIMIT 100;

* **Python:**

import pandas as pd  
  
def merge\_with\_toxicity\_data(chembl\_df, toxicity\_file):  
 """Merges ChEMBL data with external toxicity data.  
  
 Args:  
 chembl\_df (pandas.DataFrame): DataFrame containing ChEMBL data (e.g., activity, SMILES).  
 toxicity\_file (str): Path to the toxicity data file (e.g., CSV).  
  
 Returns:  
 pandas.DataFrame: Merged DataFrame.  
 """  
 toxicity\_df = pd.read\_csv(toxicity\_file) # Assuming toxicity data is in CSV  
 merged\_df = pd.merge(chembl\_df, toxicity\_df, on='molregno', how='left') # Assuming 'molregno' is the common identifier  
 return merged\_df  
  
# Example Usage:  
# toxicity\_data\_file = 'toxicity\_data.csv' # Replace with your file path  
# merged\_data = merge\_with\_toxicity\_data(df, toxicity\_data\_file)  
# print(merged\_data)

**Lưu ý quan trọng:**

* Hãy thay thế các tên target, ID hợp chất, đường dẫn file, v.v. trong các ví dụ trên bằng dữ liệu thực tế của bạn.
* Các ví dụ này chỉ là điểm khởi đầu. Bạn có thể tùy chỉnh và mở rộng chúng để phù hợp với nhu cầu cụ thể của dự án.
* Đảm bảo rằng bạn đã cài đặt tất cả các thư viện cần thiết (psycopg2, RDKit, scikit-learn, pandas, matplotlib).

Tôi hy vọng những phân tích, hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án phân tích dữ liệu ChEMBL 35 của mình một cách hiệu quả. Nếu bạn có bất kỳ câu hỏi nào hoặc cần hỗ trợ thêm, đừng ngần ngại hỏi! Chúc bạn thành công!