PharmApp Suite

🧠 AI for Drug Discovery and Development 🍼

🧩 Topic: Gộp ChEMBL với dữ liệu nhà cung cấp hóa chất

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 này. Với vai trò là một chuyên gia R&D dược học, tôi hiểu tầm quan trọng của việc khai thác thông tin từ cơ sở dữ liệu lớn như ChEMBL để tìm kiếm và phát triển các loại thuốc mới. Chúng ta sẽ cùng nhau xây dựng một quy trình phân tích hiệu quả, tuân thủ theo chuẩn AIMLOps template và giải quyết các vấn đề phát sinh.

Tổng quan dự án:

Dự án này tập trung vào việc phân tích dữ liệu từ cơ sở dữ liệu ChEMBL 35, sử dụng PostgreSQL (psql) làm hệ quản trị cơ sở dữ liệu và RDKit để xử lý thông tin hóa học. Chúng ta sẽ trích xuất dữ liệu bằng các truy vấn SQL, lưu trữ dưới dạng file CSV và sau đó sử dụng Jupyter Notebook với Python để phân tích và trực quan hóa dữ liệu.

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

Mô hình phân tích của chúng ta sẽ bao gồm các bước chính sau:

- **Data Extraction (Trích xuất dữ liệu):** Sử dụng SQL queries để trích xuất thông tin liên quan từ cơ sở dữ liệu ChEMBL 35. Chúng ta sẽ tập trung vào các bảng như `activities`, `molecule_dictionary`, `compound_structures`,...
- **Data Cleaning and Preprocessing (Làm sạch và tiền xử lý dữ liệu):**
  - Loại bỏ các giá trị bị thiếu hoặc không hợp lệ.
  - Chuyển đổi dữ liệu về định dạng phù hợp (ví dụ: chuyển đổi giá trị IC50 từ dạng chuỗi sang số).
  - Chuẩn hóa dữ liệu (ví dụ: chuẩn hóa SMILES strings).
- **Feature Engineering (Xây dựng đặc trưng):** Sử dụng RDKit để tính toán các đặc trưng hóa học từ cấu trúc phân tử (ví dụ: molecular weight, logP, số lượng hydrogen bond donors/acceptors).
- **Exploratory Data Analysis (EDA - Phân tích khám phá dữ liệu):** Sử dụng các kỹ thuật thống kê và trực quan hóa để hiểu rõ hơn về dữ liệu, tìm kiếm các mối quan hệ và xu hướng tiềm năng.
- **Modeling (Xây dựng mô hình):** Xây dựng các mô hình dự đoán (ví dụ: mô hình dự đoán hoạt tính sinh học) bằng cách sử dụng các thuật toán machine learning như linear regression, random forest, support vector machines,...
- **Model Evaluation (Đánh giá mô hình):** Đánh giá hiệu năng của mô hình bằng các metrics phù hợp (ví dụ: R-squared, RMSE, AUC).

2. Hướng dẫn song ngữ (Bilingual Guide):

2.1. Kết nối cơ sở dữ liệu PostgreSQL (Connecting to PostgreSQL Database):

- **Tiếng Việt:** Sử dụng thư viện `psycopg2` trong Python để kết nối đến cơ sở dữ liệu PostgreSQL.
- **English:** Use the `psycopg2` library in Python to connect to the PostgreSQL database.

```
# Tiếng Việt: Ví dụ kết nối cơ sở dữ liệu
# English: Example of connecting to the database
import psycopg2

try:
    conn = psycopg2.connect(
```

```
        host="192.168.206.136",
        database="chembl_35",
        user="rd",
        password="rd")

    cur = conn.cursor()
    print("Kết nối thành công đến PostgreSQL")
    print("Successfully connected to PostgreSQL")

except psycopg2.Error as e:
    print(f"Lỗi kết nối: {e}")
    print(f"Connection error: {e}")
finally:
    if conn:
        cur.close()
        conn.close()
        print("Ngắt kết nối PostgreSQL")
        print("PostgreSQL connection closed")
```

2.2. Trích xuất dữ liệu bằng SQL (Extracting Data using SQL):

- **Tiếng Việt:** Sử dụng các câu lệnh SQL `SELECT`, `JOIN`, `WHERE` để trích xuất dữ liệu từ các bảng khác nhau trong cơ sở dữ liệu ChEMBL.
- **English:** Use SQL statements like `SELECT`, `JOIN`, `WHERE` to extract data from different tables in the ChEMBL database.

```
-- Tiếng Việt: Ví dụ truy vấn SQL
-- English: Example SQL query
SELECT md.chembl_id,
       cs.canonical_smiles,
       act.standard_type,
       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
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
LIMIT 100;
```

2.3. Sử dụng RDKit để tính toán đặc trưng (Using RDKit for Feature Calculation):

- **Tiếng Việt:** Sử dụng thư viện RDKit để chuyển đổi SMILES strings thành các đối tượng phân tử và tính toán các đặc trưng hóa học.
- **English:** Use the RDKit library to convert SMILES strings into molecule objects and calculate chemical features.

```
# Tiếng Việt: Ví dụ sử dụng RDKit
# English: Example of using RDKit
from rdkit import Chem
from rdkit.Chem import Descriptors

def calculate_features(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        mw = Descriptors.MolWt(mol)
        logp = Descriptors.MolLogP(mol)
        # Tính toán thêm các đặc trưng khác
        return mw, logp
    else:
        return None, None
```

```python
# Ví dụ sử dụng hàm
# Example usage
smiles = 'CC(=O)Oc1ccccc1C(=O)O'
mw, logp = calculate_features(smiles)
print(f"Molecular Weight: {mw}, LogP: {logp}")
```

3. Code SQL, Python (English):

3.1. SQL (for data extraction):

```sql
-- SQL query to extract data for analysis
SELECT md.chembl_id,
       cs.canonical_smiles,
       act.standard_type,
       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
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL  -- Add this line to filter out NULL values
LIMIT 100;
```

Lưu ý sửa lỗi SQL: Lỗi `ERROR: operator does not exist: numeric ~ unknown` xảy ra do bạn đang cố gắng so sánh một cột kiểu số (`act.standard_value`) với một chuỗi (`'^[0-9\.]+$'`). Trong PostgreSQL, bạn không cần sử dụng regular expression để kiểm tra xem một cột kiểu số có phải là số hay không. Thay vào đó, hãy đảm bảo rằng cột `standard_value` không chứa giá trị NULL và có thể chuyển đổi thành kiểu số.

3.2. Python (for data analysis and feature engineering):

```python
import pandas as pd
import psycopg2
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np

# Database connection details
db_host = "192.168.206.136"
db_name = "chembl_35"
db_user = "rd"
db_pass = "rd"

# Function to connect to PostgreSQL
def connect_to_db():
    try:
        conn = psycopg2.connect(
            host=db_host,
            database=db_name,
            user=db_user,
            password=db_pass)
        return conn
    except psycopg2.Error as e:
        print(f"Error connecting to PostgreSQL: {e}")
        return None

# Function to execute SQL query and return data as a Pandas DataFrame
def execute_sql_query(conn, sql_query):
    try:
```

```
        df = pd.read_sql_query(sql_query, conn)
        return df
    except Exception as e:
        print(f"Error executing SQL query: {e}")
        return None

# Function to calculate RDKit features
def calculate_rdkit_features(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is not None:
            mw = Descriptors.MolWt(mol)
            logp = Descriptors.MolLogP(mol)
            hbd = Descriptors.NumHDonors(mol)
            hba = Descriptors.NumHAcceptors(mol)
            # You can add more descriptors here
            return mw, logp, hbd, hba
        else:
            return None, None, None, None
    except:
        return None, None, None, None

# Main analysis script
def main():
    # SQL query to extract data
    sql_query = """
        SELECT md.chembl_id,
               cs.canonical_smiles,
               act.standard_type,
               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
        WHERE act.standard_type = 'IC50'
          AND act.standard_units = 'nM'
          AND act.standard_value IS NOT NULL
        LIMIT 100;
    """

    # Connect to the database
    conn = connect_to_db()
    if conn is None:
        return

    # Execute the SQL query
    df = execute_sql_query(conn, sql_query)
    conn.close()

    if df is None:
        return

    # Print the first few rows of the DataFrame
    print("Original Data:")
    print(df.head())

    # Calculate RDKit features
    df[['mw', 'logp', 'hbd', 'hba']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_rdkit_features(x)))
```

```
    # Handle missing values after RDKit calculation
    df.replace([np.inf, -np.inf], np.nan, inplace=True)
    df.dropna(inplace=True)

    # Convert standard_value to numeric
    df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
    df.dropna(subset=['standard_value'], inplace=True)


    # Print the DataFrame with calculated features
    print("\nData with RDKit Features:")
    print(df.head())

    # Basic EDA (Exploratory Data Analysis)
    print("\nDescriptive Statistics:")
    print(df.describe())

if __name__ == "__main__":
    main()
```

Lưu ý về phiên bản scikit-learn: Nếu bạn gặp lỗi `TypeError: mean_squared_error() got an unexpected keyword argument 'squared'`, hãy đảm bảo rằng bạn đang sử dụng phiên bản scikit-learn mới hơn. Nếu không, bạn có thể loại bỏ tham số `squared=False` (tham số này chỉ có trong các phiên bản mới hơn).

4. Ví dụ code SQL và Python (Code Examples):

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

Ví dụ 1: Trích xuất thông tin về các hợp chất có hoạt tính trên một target cụ thể (Extracting compound information with activity on a specific target):

- **SQL:**

```
SELECT md.chembl_id,
       cs.canonical_smiles,
       act.standard_value
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'
  AND act.standard_units = 'nM'
  AND act.target_chembl_id = 'CHEMBL205' -- Replace with your target
LIMIT 100;
```

- **Python:**

```
import pandas as pd
import psycopg2

def get_compounds_by_target(target_id, limit=100):
    conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass)
    sql_query = f"""
        SELECT md.chembl_id,
               cs.canonical_smiles,
               act.standard_value
        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'
          AND act.standard_units = 'nM'
```

```
          AND act.target_chembl_id = '{target_id}'
        LIMIT {limit};
    """
    df = pd.read_sql_query(sql_query, conn)
    conn.close()
    return df

target_chembl_id = 'CHEMBL205'
compounds_df = get_compounds_by_target(target_chembl_id)
print(compounds_df.head())
```

Ví dụ 2: Tính toán số lượng hợp chất cho mỗi target (Calculate the number of compounds for each target):

- **SQL:**

```
SELECT target_chembl_id, COUNT(DISTINCT molregno) AS num_compounds
FROM activities
WHERE standard_type = 'IC50'
GROUP BY target_chembl_id
ORDER BY num_compounds DESC
LIMIT 10;
```

- **Python:**

```
import pandas as pd
import psycopg2

def count_compounds_per_target(limit=10):
    conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass)
    sql_query = f"""
        SELECT target_chembl_id, COUNT(DISTINCT molregno) AS num_compounds
        FROM activities
        WHERE standard_type = 'IC50'
        GROUP BY target_chembl_id
        ORDER BY num_compounds DESC
        LIMIT {limit};
    """
    df = pd.read_sql_query(sql_query, conn)
    conn.close()
    return df

target_counts_df = count_compounds_per_target()
print(target_counts_df.head())
```

Ví dụ 3: Phân tích phân bố giá trị IC50 (Analyze IC50 value distribution):

- **SQL:** (Không cần thiết, có thể thực hiện trực tiếp trong Python sau khi trích xuất dữ liệu)
- **Python:**

```
import pandas as pd
import psycopg2
import matplotlib.pyplot as plt
import numpy as np

def analyze_ic50_distribution(limit=100):
    conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass)
    sql_query = f"""
        SELECT standard_value
        FROM activities
        WHERE standard_type = 'IC50'
```

```
          AND standard_units = 'nM'
          AND standard_value IS NOT NULL
        LIMIT {limit};
    """
    df = pd.read_sql_query(sql_query, conn)
    conn.close()

    # Convert to numeric and drop NaN values
    df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
    df.dropna(subset=['standard_value'], inplace=True)

    # Take the logarithm of IC50 values
    df['pIC50'] = -np.log10(df['standard_value'] / 1e9)  # Convert nM to M

    plt.hist(df['pIC50'], bins=30)
    plt.xlabel('pIC50')
    plt.ylabel('Frequency')
    plt.title('Distribution of pIC50 Values')
    plt.show()

analyze_ic50_distribution()
```

Ví dụ 4: Tính toán các descriptor vật lý hóa học và xem xét tương quan (Calculate physicochemical descriptors and explore correlations):

- **SQL:** (Như ví dụ 1 để lấy SMILES)
- **Python:**

```
import pandas as pd
import psycopg2
from rdkit import Chem
from rdkit.Chem import Descriptors
import matplotlib.pyplot as plt

def analyze_descriptor_correlations(limit=100):
    conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass)
    sql_query = f"""
        SELECT 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'
          AND act.standard_units = 'nM'
        LIMIT {limit};
    """
    df = pd.read_sql_query(sql_query, conn)
    conn.close()

    def calculate_descriptors(smiles):
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol), Descriptors.MolLogP(mol)
        return None, None

    df['MW'], df['LogP'] = zip(*df['canonical_smiles'].apply(calculate_descriptors))
    df.dropna(inplace=True)

    plt.scatter(df['MW'], df['LogP'])
    plt.xlabel('Molecular Weight')
```

```
    plt.ylabel('LogP')
    plt.title('Correlation between MW and LogP')
    plt.show()

analyze_descriptor_correlations()
```

Ví dụ 5: Xây dựng mô hình dự đoán hoạt tính (Build a simple activity prediction model):

- **SQL:** (Như ví dụ 1 để lấy SMILES và IC50)
- **Python:**

```
import pandas as pd
import psycopg2
from rdkit import Chem
from rdkit.Chem import Descriptors
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

def build_activity_prediction_model(limit=100):
    conn = psycopg2.connect(host=db_host, database=db_name, user=db_user,
password=db_pass)
    sql_query = f"""
        SELECT cs.canonical_smiles, act.standard_value
        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'
          AND act.standard_units = 'nM'
          AND act.standard_value IS NOT NULL
        LIMIT {limit};
    """
    df = pd.read_sql_query(sql_query, conn)
    conn.close()

    def calculate_descriptors(smiles):
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol), Descriptors.MolLogP(mol)
        return None, None

    df['MW'], df['LogP'] = zip(*df['canonical_smiles'].apply(calculate_descriptors))
    df.dropna(inplace=True)
    df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
    df.dropna(subset=['standard_value'], inplace=True)

    X = df[['MW', 'LogP']]
    y = -np.log10(df['standard_value'] / 1e9) # pIC50

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    model = LinearRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print(f'Mean Squared Error: {mse}')
```

```
build_activity_prediction_model()
```

Lưu ý quan trọng:

- **Đường dẫn (Paths):** Luôn sử dụng `os.path.join(base_path, ...)` để đảm bảo tính tương thích của code trên các hệ điều hành khác nhau.
- **Xử lý lỗi (Error Handling):** Bổ sung các khối `try...except` để xử lý các lỗi có thể xảy ra trong quá trình kết nối cơ sở dữ liệu, tính toán đặc trưng, hoặc xây dựng mô hình.
- **Logging:** Sử dụng thư viện `logging` để ghi lại các thông tin quan trọng trong quá trình chạy code.
- **AIMLOps Template:** Đảm bảo rằng code của bạn tuân thủ theo cấu trúc thư mục và quy ước đặt tên file của AIMLOps template.

Tôi hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án của mình một cách thuận lợi. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi!