# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 💉



## 🧩 Topic: Kiểm chứng chéo tương tác với DrugBank

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 R&D dược học, tôi sẽ cung cấp hướng dẫn chi tiết, code mẫu SQL và Python, cùng những lưu ý quan trọng để bạn hoàn thành dự án một cách hiệu quả.

**Tổng quan dự án:**

Dự án của bạn tập trung vào việc phân tích dữ liệu từ cơ sở dữ liệu ChEMBL 35, sử dụng RDKit để xử lý thông tin hóa học, và tuân theo cấu trúc thư mục chuẩn AIMLOps. Mục tiêu là trích xuất thông tin hữu ích cho quá trình nghiên cứu và phát triển thuốc.

**Phân tích mô hình (Analysis Model):**

Chúng ta sẽ tập trung vào việc xây dựng một quy trình làm việc (workflow) hoàn chỉnh từ việc truy vấn dữ liệu từ ChEMBL, tiền xử lý dữ liệu bằng RDKit, và cuối cùng là phân tích dữ liệu để trả lời các câu hỏi nghiên cứu cụ thể.

**Các bước chính:**

1. **Truy vấn dữ liệu từ ChEMBL (Data Retrieval):** Sử dụng SQL để trích xuất thông tin về các hợp chất và hoạt tính sinh học của chúng.
2. **Tiền xử lý dữ liệu (Data Preprocessing):** Sử dụng RDKit để chuyển đổi SMILES thành các đặc trưng hóa học (chemical features) có thể sử dụng trong các mô hình học máy.
3. **Phân tích dữ liệu (Data Analysis):** Sử dụng các kỹ thuật thống kê và học máy để khám phá các mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học.

**Hướng dẫn song ngữ (Bilingual Instructions):**

- **English:** This project aims to analyze ChEMBL 35 data using RDKit within an AIMLOps framework. The goal is to extract valuable information for drug discovery and development. We will focus on building a complete workflow from data retrieval to analysis.
- **Tiếng Việt:** Dự án này nhằm mục đích phân tích dữ liệu ChEMBL 35 sử dụng RDKit trong khuôn khổ AIMLOps. Mục tiêu là trích xuất thông tin giá trị cho việc khám phá và phát triển thuốc. Chúng ta sẽ tập trung vào việc xây dựng một quy trình làm việc hoàn chỉnh từ việc truy xuất dữ liệu đến phân tích.

**Code SQL (SQL Code):**

Dưới đây là một số ví dụ SQL, lưu ý chỉ lấy 100 dòng để giảm tải cho máy tính.

*Ví dụ 1: Lấy thông tin cơ bản về các hợp chất có hoạt tính ức chế trên một mục tiêu cụ thể (e.g., EGFR).*

```sql
-- English
-- Retrieve basic information about compounds with inhibitory activity on a specific target (e.g., EGFR)
SELECT DISTINCT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
FROM
    molecule_dictionary md
```

```
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Epidermal Growth Factor Receptor'  -- Replace with your target of
interest
    AND act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
    AND act.standard_value ~ '^[0-9\.]+$'  -- Ensure standard_value is numeric
LIMIT 100;

-- Tiếng Việt
-- Lấy thông tin cơ bản về các hợp chất có hoạt tính ức chế trên một mục tiêu cụ thể
(ví dụ: EGFR)
SELECT DISTINCT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Epidermal Growth Factor Receptor'  -- Thay thế bằng mục tiêu bạn
quan tâm
    AND act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
    AND act.standard_value ~ '^[0-9\.]+$'  -- Đảm bảo standard_value là kiểu số
LIMIT 100;
```

*Ví dụ 2: Lấy thông tin về các hợp chất có hoạt tính trên một protein cụ thể và lưu vào file CSV.*

```
-- English
-- Retrieve information about compounds active on a specific protein and save to a CSV
file.
-- Requires you to run this in pgAdmin and export the result to a CSV file.
SELECT DISTINCT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
```

```
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Tyrosine-protein kinase ABL1'  -- Example protein target
    AND act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
    AND act.standard_value ~ '^[0-9\.]+$'  -- Ensure standard_value is numeric
LIMIT 100;

-- Tiếng Việt
-- Lấy thông tin về các hợp chất có hoạt tính trên một protein cụ thể và lưu vào file
CSV.
-- Yêu cầu bạn chạy truy vấn này trong pgAdmin và xuất kết quả ra file CSV.
SELECT DISTINCT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Tyrosine-protein kinase ABL1'  -- Ví dụ mục tiêu protein
    AND act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
    AND act.standard_value ~ '^[0-9\.]+$'  -- Đảm bảo standard_value là kiểu số
LIMIT 100;
```

*Ví dụ 3: Tìm kiếm các hợp chất có khối lượng phân tử nằm trong một khoảng nhất định.*

```
-- English
-- Find compounds with molecular weight within a certain range.
SELECT DISTINCT
    md.chembl_id,
    cs.canonical_smiles,
    md.mw_freebase
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    md.mw_freebase BETWEEN 400 AND 600
LIMIT 100;

-- Tiếng Việt
-- Tìm kiếm các hợp chất có khối lượng phân tử nằm trong một khoảng nhất định.
SELECT DISTINCT
    md.chembl_id,
    cs.canonical_smiles,
    md.mw_freebase
FROM
```

```
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    md.mw_freebase BETWEEN 400 AND 600
LIMIT 100;
```

*Ví dụ 4: Lấy thông tin về các mục tiêu và số lượng hợp chất hoạt động trên mỗi mục tiêu.*

```
-- English
-- Get information about targets and the number of compounds active on each target.
SELECT
    td.chembl_id,
    td.pref_name,
    COUNT(DISTINCT act.molregno) AS num_compounds
FROM
    target_dictionary td
JOIN
    activities act ON td.tid = act.tid
WHERE act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
    AND act.standard_value ~ '^[0-9\.]+$'  -- Ensure standard_value is numeric
GROUP BY
    td.chembl_id, td.pref_name
ORDER BY
    num_compounds DESC
LIMIT 100;

-- Tiếng Việt
-- Lấy thông tin về các mục tiêu và số lượng hợp chất hoạt động trên mỗi mục tiêu.
SELECT
    td.chembl_id,
    td.pref_name,
    COUNT(DISTINCT act.molregno) AS num_compounds
FROM
    target_dictionary td
JOIN
    activities act ON td.tid = act.tid
WHERE act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
    AND act.standard_value ~ '^[0-9\.]+$'  -- Đảm bảo standard_value là kiểu số
GROUP BY
    td.chembl_id, td.pref_name
ORDER BY
    num_compounds DESC
LIMIT 100;
```

*Ví dụ 5: Tìm kiếm các hợp chất có hoạt tính trên một mục tiêu cụ thể và có giá trị pChEMBL lớn hơn một giá trị nhất định.*

```
-- English
-- Find compounds active on a specific target and have a pChEMBL value greater than a
certain value.
SELECT DISTINCT
    md.chembl_id,
    cs.canonical_smiles,
    act.pchembl_value
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
```

```sql
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Cyclooxygenase-2'  -- Example target
    AND act.pchembl_value > 6.0
LIMIT 100;

-- Tiếng Việt
-- Tìm kiếm các hợp chất có hoạt tính trên một mục tiêu cụ thể và có giá trị pChEMBL lớn hơn một giá trị nhất định.
SELECT DISTINCT
    md.chembl_id,
    cs.canonical_smiles,
    act.pchembl_value
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Cyclooxygenase-2'  -- Ví dụ mục tiêu
    AND act.pchembl_value > 6.0
LIMIT 100;
```

**Lưu ý về lỗi SQL (Note on SQL Error):**

Lỗi `ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard_value ~ '^[0-9\.]+$'` xảy ra do kiểu dữ liệu của cột `act.standard_value` không phải là numeric. Để khắc phục, bạn có thể thử ép kiểu dữ liệu trước khi so sánh:

```sql
AND act.standard_value::numeric ~ '^[0-9\.]+$'
```

Hoặc sử dụng hàm `REGEXP_MATCHES` (tùy thuộc vào phiên bản PostgreSQL của bạn).

**Code Python (Python Code):**

Dưới đây là ví dụ code Python sử dụng RDKit để đọc file CSV, tính toán fingerprint và chuẩn bị dữ liệu cho mô hình học máy.

```python
# English
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.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Define base path (replace with your actual path)
base_path = "../data"

# Define file path for the CSV file
csv_file_path = os.path.join(base_path, "your_data.csv")  # Replace with your actual CSV file name
```

```
# Load data from CSV
try:
    data = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: The file {csv_file_path} was not found.")
    exit()

# Data Cleaning and Preprocessing
data = data.dropna(subset=['canonical_smiles', 'standard_value'])
data = data[data['standard_value'].astype(str).str.match(r'^[0-9\.]+$')]  # Keep only
numeric values
data['standard_value'] = pd.to_numeric(data['standard_value'], errors='coerce')
data = data.dropna(subset=['standard_value'])
data = data.head(100)

# RDKit Function
def generate_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
        return np.array(list(fp.ToBitString()), dtype=int)
    else:
        return None

# Generate Fingerprints
data['fingerprint'] = data['canonical_smiles'].apply(generate_fingerprint)
data = data.dropna(subset=['fingerprint'])

# Prepare Data for Machine Learning
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)

# Train Model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate Model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")


# Vietnamese
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.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Định nghĩa đường dẫn gốc (thay thế bằng đường dẫn thực tế của bạn)
base_path = "../data"
```

```
# Định nghĩa đường dẫn đến file CSV
csv_file_path = os.path.join(base_path, "your_data.csv")  # Thay thế bằng tên file CSV thực tế của bạn

# Tải dữ liệu từ file CSV
try:
    data = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Lỗi: Không tìm thấy file {csv_file_path}.")
    exit()

# Làm sạch và tiền xử lý dữ liệu
data = data.dropna(subset=['canonical_smiles', 'standard_value'])
data = data[data['standard_value'].astype(str).str.match(r'^[0-9\.]+$')]  # Chỉ giữ lại các giá trị số
data['standard_value'] = pd.to_numeric(data['standard_value'], errors='coerce')
data = data.dropna(subset=['standard_value'])
data = data.head(100)

# Hàm RDKit
def generate_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
        return np.array(list(fp.ToBitString()), dtype=int)
    else:
        return None

# Tạo Fingerprints
data['fingerprint'] = data['canonical_smiles'].apply(generate_fingerprint)
data = data.dropna(subset=['fingerprint'])

# Chuẩn bị dữ liệu cho Học Máy
X = np.stack(data['fingerprint'].values)
y = data['standard_value'].values

# Chia Dữ Liệu
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Huấn Luyện Mô Hình
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Đánh Giá Mô Hình
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Lỗi Bình Phương Trung Bình: {mse}")
```

**Lưu ý về lỗi Python (Note on Python Error):**

Lỗi về tham số `squared=False` trong `mean_squared_error` chỉ xảy ra với các phiên bản scikit-learn rất cũ. Bạn nên nâng cấp scikit-learn lên phiên bản mới nhất để tránh lỗi này. Nếu không thể nâng cấp, bạn có thể tính căn bậc hai của MSE để có RMSE (Root Mean Squared Error).

**Các ví dụ Python khác:**

*Ví dụ 1: Tính các descriptor phân tử cơ bản (ví dụ: LogP, MW) sử dụng RDKit.*

```
# English
from rdkit import Chem
```

```
from rdkit.Chem import Descriptors

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

data['logp'], data['mw'] = zip(*data['canonical_smiles'].apply(calculate_descriptors))
print(data[['canonical_smiles', 'logp', 'mw']].head())

# Vietnamese
from rdkit import Chem
from rdkit.Chem import Descriptors

def tinh_toan_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        logp = Descriptors.MolLogP(mol)
        mw = Descriptors.MolWt(mol)
        return logp, mw
    else:
        return None, None

data['logp'], data['mw'] = zip(*data['canonical_smiles'].apply(tinh_toan_descriptors))
print(data[['canonical_smiles', 'logp', 'mw']].head())
```

*Ví dụ 2: Trực quan hóa một phân tử từ SMILES.*

```
# English
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from IPython.display import SVG

def visualize_molecule(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return SVG(Draw.MolToSVG(mol, size=(300, 300)))
    else:
        return None

smiles = data['canonical_smiles'].iloc[0]
visualize_molecule(smiles)

# Vietnamese
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from IPython.display import SVG

def truc_quan_hoa_phan_tu(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return SVG(Draw.MolToSVG(mol, size=(300, 300)))
    else:
        return None
```

```
smiles = data['canonical_smiles'].iloc[0]
truc_quan_hoa_phan_tu(smiles)
```

*Ví dụ 3: Tạo và trực quan hóa fingerprint (ví dụ: Morgan fingerprint).*

```
# English
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Draw
from IPython.display import SVG

def visualize_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprint(mol, 2)
        return fp
    else:
        return None

fp = visualize_fingerprint(data['canonical_smiles'].iloc[0])
print(fp)

# Vietnamese
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Draw
from IPython.display import SVG

def truc_quan_hoa_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprint(mol, 2)
        return fp
    else:
        return None

fp = truc_quan_hoa_fingerprint(data['canonical_smiles'].iloc[0])
print(fp)
```

*Ví dụ 4: Sử dụng PCA để giảm chiều dữ liệu fingerprint.*

```
# English
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Assuming X is the fingerprint matrix from the previous example
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('PCA of Fingerprints')
plt.show()

# Vietnamese
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

```
# Giả sử X là ma trận fingerprint từ ví dụ trước
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.xlabel('Thành phần PCA 1')
plt.ylabel('Thành phần PCA 2')
plt.title('PCA của Fingerprints')
plt.show()
```

*Ví dụ 5: Xây dựng mô hình Random Forest để dự đoán hoạt tính.*

```
# English
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

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

model = RandomForestRegressor(n_estimators=100, random_state=42)
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}")

# Vietnamese
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

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

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Lỗi Bình Phương Trung Bình: {mse}")
```

**Cấu trúc thư mục (Folder Structure):**

Đảm bảo rằng bạn tuân thủ cấu trúc thư mục AIMLOps:

```
├── data/
│   └── your_data.csv  # Dữ liệu CSV từ ChEMBL
├── notebooks/
│   ├── Topic_CheMBL_35_83_1_data_extraction.ipynb
│   ├── Topic_CheMBL_35_83_2_data_analysis.ipynb
├── models/
│   └──  # Lưu trữ các mô hình đã huấn luyện
├── scripts/
│   └──  # Lưu trữ các script Python
└── README.md
```

**Lời khuyên (Advice):**

- **Chia nhỏ các bước:** Chia dự án thành các phần nhỏ hơn, dễ quản lý hơn (ví dụ: trích xuất dữ liệu, tiền xử lý, phân tích).

- **Ghi lại quá trình:** Ghi lại tất cả các bước thực hiện, các quyết định và kết quả. Điều này rất quan trọng để tái tạo và gỡ lỗi.
- **Kiểm tra dữ liệu:** Luôn kiểm tra dữ liệu của bạn để đảm bảo tính chính xác và đầy đủ.
- **Sử dụng virtual environment:** Sử dụng virtual environment để quản lý các thư viện Python và tránh xung đột.

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