PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Dashboard Streamlit để khám phá hoạt tính sinh học

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 chuyên gia nghiên cứu và phát triển dược học, tôi sẽ cung cấp hướng dẫn chi tiết, mã nguồn SQL và Python mẫu, cũng như các ví dụ cụ thể để bạn có thể triển khai dự án một cách hiệu quả.

Tổng quan dự án:

Dự án của bạn tập trung vào việc khai thác dữ liệu từ cơ sở dữ liệu ChEMBL 35 kết hợp với các công cụ tin sinh học như RDKit để hỗ trợ quá trình nghiên cứu và phát triển thuốc. Việc sử dụng chuẩn AIMLOps Template giúp đảm bảo tính tổ chức và khả năng tái sử dụng của dự án.

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

Mục tiêu của bạn là phân tích dữ liệu ChEMBL 35 để tìm ra các mối liên hệ giữa cấu trúc hóa học của các hợp chất và hoạt tính sinh học của chúng. Để đạt được điều này, bạn có thể áp dụng mô hình phân tích sau:

- **Thu thập và tiền xử lý dữ liệu (Data Acquisition and Preprocessing):**
    - Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35, tập trung vào các thông tin như cấu trúc phân tử (ví dụ: SMILES), hoạt tính sinh học (ví dụ: IC50, Ki) và các thuộc tính liên quan khác.
    - Tiền xử lý dữ liệu bằng RDKit để chuyển đổi cấu trúc SMILES thành các descriptor (ví dụ: fingerprints, physicochemical properties) có thể sử dụng trong các mô hình học máy.
- **Phân tích khám phá dữ liệu (Exploratory Data Analysis - EDA):**
    - Sử dụng các kỹ thuật thống kê và trực quan hóa để khám phá các đặc trưng của dữ liệu, tìm kiếm các xu hướng, mối tương quan và các điểm bất thường.
    - Ví dụ: phân tích phân bố của các giá trị hoạt tính, mối tương quan giữa các descriptor khác nhau, hoặc sự khác biệt về cấu trúc giữa các hợp chất có hoạt tính cao và thấp.
- **Xây dựng mô hình học máy (Machine Learning Model Building):**
    - Sử dụng các thuật toán học máy như hồi quy tuyến tính, random forest, support vector machines (SVM) hoặc mạng nơ-ron để xây dựng mô hình dự đoán hoạt tính dựa trên các descriptor đã tính toán.
    - Đánh giá hiệu suất của mô hình bằng các kỹ thuật validation phù hợp (ví dụ: cross-validation) và các metrics như R-squared, RMSE, AUC.
- **Giải thích và ứng dụng kết quả (Interpretation and Application of Results):**
    - Phân tích các yếu tố quan trọng nhất ảnh hưởng đến hoạt tính sinh học, từ đó đưa ra các gợi ý về thiết kế và tối ưu hóa cấu trúc phân tử.
    - Sử dụng mô hình để dự đoán hoạt tính của các hợp chất mới, sàng lọc ảo (virtual screening) và xác định các ứng cử viên tiềm năng cho phát triển thuốc.

2. Hướng dẫn song ngữ (Bilingual Instructions):

English:

1. **Data Extraction (SQL):** Use SQL queries to extract relevant data from the ChEMBL 35 database, focusing on molecular structures (SMILES), bioactivity data (IC50, Ki), and other relevant properties.

2. **Data Preprocessing (Python with RDKit):** Use RDKit to convert SMILES strings into molecular descriptors (fingerprints, physicochemical properties) suitable for machine learning models.
3. **Exploratory Data Analysis (Python):** Perform statistical analysis and visualization to explore data characteristics, identify trends, correlations, and outliers.
4. **Model Building (Python with Scikit-learn):** Build machine learning models to predict bioactivity based on calculated descriptors. Evaluate model performance using appropriate validation techniques and metrics.
5. **Interpretation and Application:** Analyze the most important factors influencing bioactivity, provide insights for molecular design and optimization, and use the model for virtual screening and candidate identification.

Tiếng Việt:

1. **Trích xuất dữ liệu (SQL):** Sử dụng truy vấn SQL để trích xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL 35, tập trung vào cấu trúc phân tử (SMILES), dữ liệu hoạt tính sinh học (IC50, Ki) và các thuộc tính liên quan khác.
2. **Tiền xử lý dữ liệu (Python với RDKit):** Sử dụng RDKit để chuyển đổi chuỗi SMILES thành các descriptor phân tử (fingerprints, thuộc tính lý hóa) phù hợp cho các mô hình học máy.
3. **Phân tích khám phá dữ liệu (Python):** Thực hiện phân tích thống kê và trực quan hóa để khám phá các đặc trưng của dữ liệu, xác định xu hướng, tương quan và các điểm ngoại lệ.
4. **Xây dựng mô hình (Python với Scikit-learn):** Xây dựng các mô hình học máy để dự đoán hoạt tính sinh học dựa trên các descriptor đã tính toán. Đánh giá hiệu suất mô hình bằng các kỹ thuật validation và metrics phù hợp.
5. **Giải thích và ứng dụng:** Phân tích các yếu tố quan trọng nhất ảnh hưởng đến hoạt tính sinh học, đưa ra các gợi ý cho thiết kế và tối ưu hóa phân tử, và sử dụng mô hình để sàng lọc ảo và xác định các ứng cử viên tiềm năng.

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

SQL (để trích xuất dữ liệu từ ChEMBL):

```sql
-- Lấy 100 dòng dữ liệu từ bảng compound_structures và activities
SELECT
    cs.molregno,
    cs.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units
FROM
    compound_structures cs
JOIN
    activities act ON cs.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
LIMIT 100;
```

Python (để tiền xử lý dữ liệu và tính toán descriptor):

```python
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
import numpy as np

base_path = "."  # Thay đổi nếu cần
```

```python
# Đọc dữ liệu từ file CSV
data_path = os.path.join(base_path, "data", "chembl_data.csv")
df = pd.read_csv(data_path)

# Hàm tính toán fingerprints sử dụng RDKit
def calculate_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)
        return np.array(fp)
    else:
        return None

# Áp dụng hàm tính toán fingerprint cho cột 'canonical_smiles'
df['fingerprint'] = df['canonical_smiles'].apply(calculate_fingerprint)

# Loại bỏ các dòng có fingerprint là None
df = df.dropna(subset=['fingerprint'])

# In ra số lượng dòng còn lại và một vài dòng đầu tiên
print(f"Số lượng dòng sau khi loại bỏ None: {len(df)}")
print(df.head())
```

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

Ví dụ 1: SQL - Lọc các hợp chất có hoạt tính IC50 dưới 100 nM:

```sql
SELECT
    cs.canonical_smiles,
    act.standard_value
FROM
    compound_structures cs
JOIN
    activities act ON cs.molregno = act.molregno
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value < 100
LIMIT 100;
```

Ví dụ 1: Python - Tính toán logP và MW sử dụng RDKit:

```python
from rdkit.Chem import Descriptors

def calculate_logp_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        logp = Descriptors.MolLogP(mol)
        mw = Descriptors.MolWt(mol)
        return logp, mw
    else:
        return None, None

df['logp'], df['mw'] = zip(*df['canonical_smiles'].apply(calculate_logp_mw))
df = df.dropna(subset=['logp', 'mw'])
print(df[['canonical_smiles', 'logp', 'mw']].head())
```

Ví dụ 2: SQL - Tìm các hợp chất tác động lên một target cụ thể (ví dụ: kinase):

```sql
SELECT
    cs.canonical_smiles,
    act.standard_value
```

```sql
FROM
    compound_structures cs
JOIN
    activities act ON cs.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE td.target_name LIKE '%Kinase%'
  AND act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
LIMIT 100;
```

Ví dụ 2: Python - Chuẩn hóa giá trị IC50 (pIC50):

```python
def calculate_pic50(ic50):
    # IC50 phải ở đơn vị nM
    if ic50 is not None:
        pic50 = -np.log10(ic50 / 1e9)  # Chuyển nM về M
        return pic50
    else:
        return None

df['pIC50'] = df['standard_value'].apply(calculate_pic50)
df = df.dropna(subset=['pIC50'])
print(df[['standard_value', 'pIC50']].head())
```

Ví dụ 3: SQL - Lấy dữ liệu về các hợp chất có hoạt tính trên một protein cụ thể (sử dụng accession number):

```sql
SELECT
    cs.canonical_smiles,
    act.standard_value
FROM
    compound_structures cs
JOIN
    activities act ON cs.molregno = act.molregno
JOIN
    target_components tc ON act.tid = tc.tid
WHERE tc.accession = 'P00533' -- Ví dụ: EGFR
  AND act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
LIMIT 100;
```

Ví dụ 3: Python - Sử dụng PCA để giảm chiều dữ liệu fingerprint:

```python
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Chuẩn hóa dữ liệu
X = np.array(list(df['fingerprint']))
X_scaled = StandardScaler().fit_transform(X)

# Áp dụng PCA
pca = PCA(n_components=10)  # Giữ lại 10 thành phần chính
X_pca = pca.fit_transform(X_scaled)

print(X_pca.shape)
```

Ví dụ 4: SQL - Tìm các hợp chất có cấu trúc tương tự với một hợp chất đã biết (sử dụng SMILES):

```sql
-- Cần có module pg_trgm để sử dụng similarity
CREATE EXTENSION IF NOT EXISTS pg_trgm;
```

```sql
SELECT
    cs.canonical_smiles,
    similarity(cs.canonical_smiles, 'Cc1ccccc1') AS sml
FROM
    compound_structures cs
WHERE cs.canonical_smiles % 'Cc1ccccc1'
ORDER BY sml DESC
LIMIT 100;
```

Ví dụ 4: Python - Xây dựng mô hình Random Forest để dự đoán pIC50:

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Chuẩn bị dữ liệu
X = np.array(list(df['fingerprint']))
y = df['pIC50']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Xây dựng mô hình Random Forest
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Dự đoán và đánh giá mô hình
y_pred = model.predict(X_test)
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}')
```

Ví dụ 5: SQL - Thống kê số lượng hợp chất cho mỗi giá trị standard_type:

```sql
SELECT act.standard_type, COUNT(*)
FROM activities act
GROUP BY act.standard_type
ORDER BY COUNT(*) DESC;
```

Ví dụ 5: Python - Trực quan hóa kết quả dự đoán mô hình:

```python
import matplotlib.pyplot as plt

plt.scatter(y_test, y_pred)
plt.xlabel("Giá trị thực tế (pIC50)")
plt.ylabel("Giá trị dự đoán (pIC50)")
plt.title("Giá trị thực tế so với giá trị dự đoán")

# Vẽ đường thẳng y=x để so sánh
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)

plt.show()
```

Giải quyết các lỗi:

- **Lỗi a: `ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`**

- **Nguyên nhân:** Lỗi này xảy ra do bạn đang cố gắng sử dụng toán tử ~ (regex match) trên một cột kiểu số (`numeric`).
- **Giải pháp:** Thay vì sử dụng regex, bạn có thể sử dụng hàm `IS NUMERIC()` hoặc ép kiểu dữ liệu sang text trước khi sử dụng regex:

```sql
-- Sử dụng IS NUMERIC() (có thể không hoạt động trên mọi phiên bản PostgreSQL)
SELECT * FROM activities WHERE IS_NUMERIC(act.standard_value);

-- Ép kiểu sang text trước khi sử dụng regex
SELECT * FROM activities WHERE CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$';
```

- **Lỗi b: `phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean_squared_error`**
    - **Nguyên nhân:** Phiên bản scikit-learn bạn đang sử dụng quá cũ và không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`. Tham số này được sử dụng để trả về root mean squared error (RMSE) thay vì mean squared error (MSE).
    - **Giải pháp:** Có hai cách để giải quyết vấn đề này:
        1. **Nâng cấp scikit-learn:** Cập nhật lên phiên bản scikit-learn mới nhất bằng lệnh `pip install --upgrade scikit-learn`.
        2. **Tính RMSE thủ công:** Nếu không thể nâng cấp scikit-learn, bạn có thể tính RMSE bằng cách lấy căn bậc hai của MSE:

```python
from sklearn.metrics import mean_squared_error
import numpy as np

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f'RMSE: {rmse}')
```

Lưu ý quan trọng:

- Đảm bảo rằng bạn đã cài đặt đầy đủ các thư viện cần thiết như RDKit, pandas, scikit-learn, và matplotlib.
- Điều chỉnh đường dẫn file và tên bảng cho phù hợp với cấu trúc dự án của bạn.
- Thử nghiệm với các thuật toán học máy khác nhau và các tham số khác nhau để tìm ra mô hình tốt nhất cho dữ liệu của bạn.
- Chú ý đến việc đánh giá mô hình một cách khách quan bằng cách sử dụng các kỹ thuật validation phù hợp.

Hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn triển khai dự án phân tích dữ liệu ChEMBL 35 một cách thành công! Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi.