PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: 50 protein mục tiêu có nhiều hoạt tính nhất

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, mã SQL và Python mẫu, cùng với các giải thích để bạn có thể thực hiện dự án một cách hiệu quả.

Tổng quan dự án

Mục tiêu của dự án này là khai thác dữ liệu từ cơ sở dữ liệu ChEMBL 35, kết hợp với các công cụ tin học hóa học (chemoinformatics) như RDKit, để hỗ trợ quá trình nghiên cứu và phát triển thuốc. Cụ thể, chúng ta sẽ tập trung vào việc:

1. **Trích xuất dữ liệu:** Sử dụng SQL để truy vấn và trích xuất thông tin liên quan từ cơ sở dữ liệu ChEMBL 35.
2. **Xử lý và làm sạch dữ liệu:** Sử dụng Python và RDKit để xử lý, làm sạch và chuẩn hóa dữ liệu, bao gồm cả việc chuyển đổi SMILES thành fingerprint phân tử.
3. **Phân tích dữ liệu:** Áp dụng các kỹ thuật phân tích dữ liệu và học máy để khám phá các mối quan hệ giữa cấu trúc phân tử và hoạt tính sinh học.
4. **Xây dựng mô hình:** Xây dựng các mô hình dự đoán hoạt tính sinh học dựa trên cấu trúc phân tử.

1. Phân tích mô hình phân tích

Mô hình phân tích của chúng ta sẽ tuân theo quy trình sau:

1. **Data Extraction (Trích xuất dữ liệu):**
    - **SQL Queries:** Sử dụng các truy vấn SQL để lấy dữ liệu từ cơ sở dữ liệu ChEMBL 35. Dữ liệu này có thể bao gồm thông tin về các hợp chất, hoạt tính sinh học, mục tiêu, v.v.
    - **Data Export:** Lưu dữ liệu trích xuất vào các file CSV để sử dụng trong các bước tiếp theo.
2. **Data Preprocessing (Tiền xử lý dữ liệu):**
    - **Data Cleaning:** Loại bỏ các giá trị thiếu, xử lý các giá trị ngoại lệ và chuẩn hóa dữ liệu.
    - **Feature Engineering:** Sử dụng RDKit để tạo ra các đặc trưng phân tử (molecular features) từ cấu trúc SMILES, như fingerprints, descriptors, v.v.
3. **Data Analysis and Modeling (Phân tích dữ liệu và xây dựng mô hình):**
    - **Exploratory Data Analysis (EDA):** Thực hiện phân tích khám phá dữ liệu để hiểu rõ hơn về dữ liệu, tìm kiếm các mối quan hệ và xu hướng.
    - **Model Building:** Xây dựng các mô hình học máy để dự đoán hoạt tính sinh học dựa trên các đặc trưng phân tử. Các mô hình có thể bao gồm:
        - Linear Regression
        - Random Forest
        - Support Vector Machines (SVM)
        - Neural Networks
    - **Model Evaluation:** Đánh giá hiệu suất của các mô hình bằng cách sử dụng các metrics phù hợp, như RMSE, R-squared, AUC, v.v.

2. Hướng dẫn song ngữ

2.1. SQL (Structured Query Language)

- **Purpose:** To extract data from the ChEMBL 35 database.
- **Mục đích:** Trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL 35.

Example:

```sql
-- English: Select compound structures and bioactivity data for a specific target.
-- Tiếng Việt: Chọn cấu trúc hợp chất và dữ liệu hoạt tính sinh học cho một mục tiêu cụ thể.
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
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Acetylcholinesterase'
    AND act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND act.standard_units = 'nM'
LIMIT 100;
```

Explanation:

- `SELECT`: Specifies the columns to retrieve.
- `FROM`: Specifies the tables to retrieve data from.
- `JOIN`: Combines rows from different tables based on a related column.
- `WHERE`: Filters the rows based on specified conditions.
- `LIMIT`: giới hạn số dòng trả về
- `md.molregno`: Molecular Registry Number (Số đăng ký phân tử)
- `cs.canonical_smiles`: SMILES string đại diện cho cấu trúc phân tử
- `act.standard_type`: Loại hoạt tính sinh học (ví dụ: IC50, Ki)
- `act.standard_value`: Giá trị hoạt tính sinh học
- `act.standard_units`: Đơn vị của giá trị hoạt tính sinh học
- `td.pref_name`: Tên mục tiêu (ví dụ: Acetylcholinesterase)

2.2. Python

- **Purpose:** To process data, generate molecular features, and build predictive models.
- **Mục đích:** Xử lý dữ liệu, tạo ra các đặc trưng phân tử và xây dựng các mô hình dự đoán.

Example:

```python
# English: Import libraries and load data.
# Tiếng Việt: Nhập các thư viện và tải dữ liệu.
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, r2_score
import os

# Define base path
base_path = ".." # Assuming the notebook is one level deep

# Load data
data_path = os.path.join(base_path, "data", "acetylcholinesterase_ic50_100.csv") #
Replace with your CSV file
df = pd.read_csv(data_path)

# English: Convert SMILES to molecular fingerprints.
# Tiếng Việt: Chuyển đổi SMILES thành fingerprint phân tử.
def generate_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

df['fingerprint'] = df['canonical_smiles'].apply(generate_fingerprint)
df = df.loc[(df['fingerprint'].notnull())]

# English: Prepare data for modeling.
# Tiếng Việt: Chuẩn bị dữ liệu cho việc xây dựng mô hình.
X = np.array(list(df['fingerprint']))
y = df['standard_value'].astype(float)  # Convert to float

# English: Split data into training and test sets.
# Tiếng Việt: 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)

# English: Train a Random Forest model.
# Tiếng Việt: Huấn luyện mô hình Random Forest.
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# English: Make predictions and evaluate the model.
# Tiếng Việt: Dự đoán và đánh giá mô hình.
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse}")
print(f"R^2: {r2}")
```

Explanation:

- `pandas`: For data manipulation and analysis.
- `rdkit`: For chemoinformatics tasks, such as converting SMILES to molecular fingerprints.
- `scikit-learn`: For building and evaluating machine learning models.
- `train_test_split`: Splits the data into training and testing sets.
- `RandomForestRegressor`: A machine learning model for regression tasks.

- `mean_squared_error`: Calculates the mean squared error between the predicted and actual values.
- `r2_score`: Calculates the R-squared value, which represents the proportion of variance in the dependent variable that is predictable from the independent variables.
- The function `generate_fingerprint` converts SMILES strings to molecular fingerprints using RDKit.
- The code trains a Random Forest model on the training data and evaluates its performance on the test data.

3. 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: Lọc các hợp chất có hoạt tính IC50 với một protein mục tiêu cụ thể (SQL)

```sql
-- English: Select compounds with IC50 values for a specific target protein.
-- Tiếng Việt: Chọn các hợp chất có giá trị IC50 cho một protein 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
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'HIV-1 reverse transcriptase'
    AND act.standard_type = 'IC50'
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
    AND act.standard_units = 'nM'
LIMIT 100;
```

Ví dụ 1: Tính toán các đặc trưng phân tử cơ bản (Python)

```python
# English: Calculate basic molecular properties using RDKit.
# Tiếng Việt: Tính toán các thuộc tính phân tử cơ bản bằng RDKit.
from rdkit import Chem
from rdkit.Chem import Descriptors

def calculate_molecular_properties(smiles):
    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)
        return mw, logp, hbd, hba
    else:
        return None, None, None, None

df[['mol_weight', 'logp', 'hbd', 'hba']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_molecular_properties(x)))
```

Ví dụ 2: Lọc các hợp chất dựa trên khoảng giá trị hoạt tính (SQL)

```sql
-- English: Select compounds with activity values within a specific range.
-- Tiếng Việt: Chọn các hợp chất có giá trị hoạt tính trong một khoảng 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_relation = '='
    AND act.standard_value BETWEEN 100 AND 1000
    AND act.standard_units = 'nM'
LIMIT 100;
```

Ví dụ 2: Xây dựng mô hình hồi quy tuyến tính (Python)

```python
# English: Build a linear regression model.
# Tiếng Việt: Xây dựng mô hình hồi quy tuyến tính.
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Assuming X and y are already defined
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)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse}")
print(f"R^2: {r2}")
```

Ví dụ 3: Truy vấn dữ liệu mục tiêu và thành phần (SQL)

```sql
-- English: Retrieve target and component information.
-- Tiếng Việt: Truy xuất thông tin mục tiêu và thành phần.
SELECT
    td.tid,
    td.pref_name,
    cmp.component_id,
    cmp.component_type
FROM
    target_dictionary td
JOIN
    target_components tc ON td.tid = tc.tid
JOIN
    component_sequences cmp ON tc.component_id = cmp.component_id
WHERE
    td.target_type = 'SINGLE PROTEIN'
LIMIT 100;
```

Ví dụ 3: Sử dụng PCA để giảm số chiều dữ liệu (Python)

```python
# English: Use PCA for dimensionality reduction.
# Tiếng Việt: Sử dụng PCA để giảm số chiều dữ liệu.
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Assuming X is already defined
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=100)  # Reduce to 100 components
X_pca = pca.fit_transform(X_scaled)

print(f"Original shape: {X.shape}")
print(f"PCA shape: {X_pca.shape}")
```

Ví dụ 4: Truy vấn các hoạt chất dựa trên số lượng vòng (SQL)

```sql
-- English: Retrieve active compounds based on the number of rings.
-- Tiếng Việt: Truy xuất các hợp chất có hoạt tính dựa trên số lượng vòng.
SELECT
    md.molregno,
    cs.canonical_smiles
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    md.num_rings >= 3
LIMIT 100;
```

Ví dụ 4: Huấn luyện mô hình SVM (Python)

```python
# English: Train a Support Vector Machine (SVM) model.
# Tiếng Việt: Huấn luyện mô hình Support Vector Machine (SVM).
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# Assuming X and y are already defined
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = SVR(kernel='rbf')  # You can also try 'linear', 'poly', 'sigmoid'
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse}")
print(f"R^2: {r2}")
```

Ví dụ 5: Tìm kiếm các hợp chất có khung phân tử tương tự (SQL)

```sql
-- English: Find compounds with similar molecular frameworks.
-- Tiếng Việt: Tìm các hợp chất có khung phân tử tương tự.
SELECT
    md1.molregno,
    cs1.canonical_smiles,
    md2.molregno,
    cs2.canonical_smiles
FROM
    molecule_dictionary md1
JOIN
    compound_structures cs1 ON md1.molregno = cs1.molregno
JOIN
    molecule_dictionary md2 ON md1.molregno != md2.molregno  -- Avoid self-comparison
JOIN
    compound_structures cs2 ON md2.molregno = cs2.molregno
WHERE
    cs1.standard_inchi_key LIKE cs2.standard_inchi_key || '%'  -- Check for similar
InChI keys
LIMIT 100;
```

Ví dụ 5: Sử dụng T-SNE để trực quan hóa dữ liệu (Python)

```python
# English: Use T-SNE for data visualization.
# Tiếng Việt: Sử dụng T-SNE để trực quan hóa dữ liệu.
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Assuming X is already defined
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X)

plt.scatter(X_tsne[:, 0], X_tsne[:, 1])
plt.title('T-SNE Visualization')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.show()
```

Lưu ý về các lỗi bạn đã đề cập:

- **Lỗi SQL "operator does not exist: numeric ~ unknown"**: Lỗi này xảy ra khi bạn cố gắng sử dụng toán tử ~ (regular expression match) trên một cột kiểu số. Để khắc phục, bạn có thể ép kiểu cột `act.standard_value` về kiểu text trước khi so sánh, hoặc sử dụng các hàm số học để lọc dữ liệu.

  ```sql
  -- Example: Ép kiểu cột standard_value về text
  AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'
  ```

- **Lỗi Python "squared=False"**: Nếu bạn đang sử dụng phiên bản scikit-learn cũ, bạn có thể loại bỏ tham số `squared=False` khỏi hàm `mean_squared_error`. Nếu không, hãy nâng cấp scikit-learn lên phiên bản mới nhất.

Lưu ý quan trọng:

- **Đường dẫn:** Hãy đảm bảo rằng bạn sử dụng `os.path.join(base_path, ...)` để xây dựng đường dẫn đến các file dữ liệu và các file khác trong dự án.
- **Giới hạn dữ liệu:** Để tránh tình trạng quá tải máy tính, bạn nên bắt đầu với một tập dữ liệu nhỏ (ví dụ: 100 dòng) và tăng dần kích thước khi cần thiết.

- **Mô hình hóa:** Hãy thử nghiệm với nhiều mô hình học máy khác nhau và đánh giá hiệu suất của chúng bằng cách sử dụng các metrics phù hợp.
- **Giải thích:** Hãy giải thích rõ ràng các bước trong quy trình phân tích của bạn và kết quả bạn thu được.
- **Tài liệu:** Hãy tham khảo tài liệu của RDKit và scikit-learn để biết thêm thông tin chi tiết về các hàm và lớp bạn đang sử dụng.

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