PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: So sánh tương đồng giữa ChEMBL và BindingDB

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 nghiên cứu và phát triển dược học, tôi sẽ cung cấp cho bạn những phân tích, hướng dẫn và code mẫu để giúp bạn đạt được mục tiêu nghiên cứu của mình.

1. Phân tích Mô hình Phân tích (Analysis of Analytical Model)

Dựa trên thông tin bạn cung cấp, chúng ta sẽ xây dựng một quy trình phân tích dữ liệu ChEMBL 35 sử dụng RDKit theo các bước sau:

- **Data Extraction (Trích xuất Dữ liệu):** Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35 (PostgreSQL) và lưu kết quả vào file CSV.
- **Data Preprocessing (Tiền xử lý Dữ liệu):**
    - Đọc dữ liệu từ file CSV vào Jupyter Notebook.
    - Làm sạch dữ liệu: Xử lý giá trị thiếu, loại bỏ các bản ghi không hợp lệ.
    - Chuyển đổi dữ liệu: Chuyển đổi SMILES thành các descriptor hóa học sử dụng RDKit.
- **Feature Engineering (Xây dựng Đặc trưng):** Tạo các đặc trưng hóa học từ SMILES sử dụng RDKit (ví dụ: Molecular Weight, LogP, Hydrogen Bond Donors/Acceptors, Topological Polar Surface Area (TPSA), v.v.).
- **Model Building (Xây dựng Mô hình):** Xây dựng các mô hình học máy để dự đoán hoạt tính của các hợp chất (ví dụ: Regression models, Classification models).
- **Model Evaluation (Đánh giá Mô hình):** Đánh giá hiệu suất của mô hình bằng các metrics phù hợp (ví dụ: R-squared, RMSE, MAE cho regression; Accuracy, Precision, Recall, F1-score cho classification).
- **Visualization and Interpretation (Trực quan hóa và Giải thích):** Trực quan hóa dữ liệu và kết quả mô hình để hiểu rõ hơn về mối quan hệ giữa cấu trúc hóa học và hoạt tính.

2. Hướng dẫn Song ngữ (Bilingual Guide)

2.1. SQL (English & Vietnamese)

- **English:** SQL (Structured Query Language) is used to extract data from the ChEMBL 35 database.
- **Vietnamese:** SQL (Ngôn ngữ truy vấn có cấu trúc) được sử dụng để trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL 35.

2.2. Python (English & Vietnamese)

- **English:** Python, along with libraries like RDKit, Pandas, and Scikit-learn, is used for data preprocessing, feature engineering, model building, and evaluation.
- **Vietnamese:** Python, cùng với các thư viện như RDKit, Pandas và Scikit-learn, được sử dụng để tiền xử lý dữ liệu, xây dựng đặc trưng, xây dựng mô hình và đánh giá.

3. Code SQL, Python (English)

3.1. SQL Code (English)

```
-- Select 100 rows of data from the activities table in the ChEMBL 35 database.
-- Filter for specific activity type and standard units.
```

```sql
SELECT act.molregno, act.standard_value, act.standard_units, act.standard_type,
md.chembl_id,
       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'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0
LIMIT 100;
```

3.2. Python Code (English)

```python
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.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import os

# Define the base path
base_path = "."  # Current directory

# Load data from CSV file
data_path = os.path.join(base_path, "data", "chembl_data.csv")
df = pd.read_csv(data_path)

# Function to calculate molecular descriptors using RDKit
def calculate_descriptors(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None
        descriptors = {
            "MolWt": Descriptors.MolWt(mol),
            "LogP": Descriptors.MolLogP(mol),
            "HBD": Descriptors.NumHDonors(mol),
            "HBA": Descriptors.NumHAcceptors(mol),
            "TPSA": Descriptors.TPSA(mol),
        }
        return descriptors
    except:
        return None

# Apply the function to calculate descriptors
df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors'])

# Convert descriptors to individual columns
df = pd.concat([df, df['descriptors'].apply(pd.Series)], axis=1)
df = df.drop('descriptors', axis=1)

# Prepare data for modeling
X = df[['MolWt', 'LogP', 'HBD', 'HBA', 'TPSA']]
y = np.log10(df['standard_value'])  # Transform IC50 values
```

```
# Data scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
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}")
```

4. Ví dụ Code SQL và Python mẫu (Code Examples)

Dưới đây là 5 ví dụ về code SQL và Python mẫu, tập trung vào các khía cạnh khác nhau của phân tích dữ liệu ChEMBL 35:

4.1. Example 1: Selecting Data and Calculating Basic Descriptors

- **SQL:**

```
-- Select Chembl ID, SMILES, and Molecular Weight for compounds with IC50 values less
than 100 nM.
SELECT md.chembl_id, cs.canonical_smiles
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_units = 'nM'
  AND act.standard_value < 100
LIMIT 100;
```

- **Python:**

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

# Load data (replace 'your_data.csv' with your actual file path)
data = pd.read_csv('your_data.csv')

# Function to calculate molecular weight
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolWt(mol)
    return None

# Apply the function to the 'SMILES' column
data['MolWt'] = data['canonical_smiles'].apply(calculate_mw)
```

```
# Print the first few rows with the calculated molecular weight
print(data.head())
```

4.2. Example 2: Filtering Data Based on Molecular Properties

- **SQL:**

```
-- Select compounds with LogP values between 2 and 4.
-- (Note: This requires a table with pre-calculated LogP values)
-- Assuming you have a table called 'molecular_properties' with 'molregno' and 'logp'
columns
SELECT md.chembl_id, cs.canonical_smiles
FROM molecule_dictionary md
JOIN compound_structures cs ON md.molregno = cs.molregno
JOIN molecular_properties mp ON md.molregno = mp.molregno
WHERE mp.logp BETWEEN 2 AND 4
LIMIT 100;
```

- **Python:**

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

# Load data
data = pd.read_csv('your_data.csv')

# Function to calculate LogP
def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolLogP(mol)
    return None

# Calculate LogP
data['LogP'] = data['canonical_smiles'].apply(calculate_logp)

# Filter data
filtered_data = data[(data['LogP'] >= 2) & (data['LogP'] <= 4)]

# Print the first few rows of the filtered data
print(filtered_data.head())
```

4.3. Example 3: Building a Simple Regression Model

- **SQL:** (No SQL needed for model building, this is done in Python)
- **Python:**

```
import pandas as pd
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

# Load data
data = pd.read_csv('your_data.csv')

# Calculate descriptors
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
```

```
    if mol:
        return Descriptors.MolWt(mol), Descriptors.MolLogP(mol)
    return None, None

data[['MolWt', 'LogP']] = data['canonical_smiles'].apply(lambda x:
pd.Series(calculate_descriptors(x)))
data = data.dropna()

# Prepare data
X = data[['MolWt', 'LogP']]
y = data['standard_value'] #IC50

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

# Predict and evaluate
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```

4.4. Example 4: Calculating TPSA (Topological Polar Surface Area)

- **SQL:** (Similar to LogP, requires pre-calculated TPSA values or a custom function in SQL, which is less common)
- **Python:**

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

# Load data
data = pd.read_csv('your_data.csv')

# Function to calculate TPSA
def calculate_tpsa(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.TPSA(mol)
    return None

# Apply the function
data['TPSA'] = data['canonical_smiles'].apply(calculate_tpsa)

# Print the first few rows
print(data.head())
```

4.5. Example 5: Handling Errors in SMILES Parsing

- **SQL:** (Focus on data selection, error handling is primarily in Python)
- **Python:**

```
import pandas as pd
from rdkit import Chem
```

```
from rdkit.Chem import Descriptors

# Load data
data = pd.read_csv('your_data.csv')

# Function to calculate molecular weight with error handling
def calculate_mw_safe(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol)
        else:
            return None  # Handle invalid SMILES
    except Exception as e:
        print(f"Error processing SMILES: {smiles}, Error: {e}")
        return None

# Apply the function
data['MolWt'] = data['canonical_smiles'].apply(calculate_mw_safe)

# Remove rows with invalid SMILES
data = data.dropna(subset=['MolWt'])

# Print the first few rows
print(data.head())
```

5. Giải quyết các vấn đề (Addressing Errors)

- **Error a:** `ERROR: operator does not exist: numeric ~ unknown LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`
    - **Explanation:** Lỗi này xảy ra vì bạn đang cố gắng sử dụng toán tử ~ (regular expression match) trên một cột kiểu số (numeric).
    - **Solution:**
        - **Cách 1:** Ép kiểu `standard_value` sang kiểu text trước khi so sánh:

```
AND TEXT(act.standard_value) ~ '^[0-9\.]+$'
```

        - **Cách 2:** Loại bỏ điều kiện này nếu bạn chỉ muốn lọc các giá trị số dương:

```
AND act.standard_value > 0
```

- **Error b:** Phiên bản scikit-learn cũ không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`
    - **Explanation:** Tham số `squared=False` được thêm vào hàm `mean_squared_error` trong các phiên bản scikit-learn mới hơn để trả về Root Mean Squared Error (RMSE) thay vì Mean Squared Error (MSE).
    - **Solution:**
        - **Cách 1:** Nâng cấp scikit-learn lên phiên bản mới nhất:

```
pip install --upgrade scikit-learn
```

        - **Cách 2:** Tính RMSE thủ công:

```
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"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
```

Lưu ý quan trọng:

- **Đường dẫn file:** Luôn sử dụng `os.path.join(base_path, ...)` để đảm bảo tính tương thích của code trên các hệ điều hành khác nhau.
- **Kiểm tra dữ liệu:** Thường xuyên kiểm tra dữ liệu sau mỗi bước để đảm bảo tính chính xác và phát hiện các vấn đề tiềm ẩn.
- **Logging:** Sử dụng logging để ghi lại các thông tin quan trọng trong quá trình chạy code, giúp bạn dễ dàng debug khi có lỗi xảy ra.

Tôi hy vọng những phân tích, hướng dẫn và code mẫu này sẽ giúp bạn trong dự án phân tích dữ liệu ChEMBL 35. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi!