PharmApp Suite

🧠 AI for Drug Discovery and Development 🧪

🧩 Topic: Tìm kiếm tương đồng Tanimoto với ChEMBL và RDKit

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 bằng RDKit. Dưới đây là phân tích chi tiết, hướng dẫn, và các ví dụ code để bạn bắt đầu, tập trung vào chủ đề **Topic_CheMBL_35_9**.

1. Phân tích và Mô hình Phân tích (Analysis and Analysis Model)

Chủ đề (Topic): Topic_CheMBL_35_9 có thể liên quan đến một mục tiêu cụ thể (ví dụ: một protein, một loại enzyme) hoặc một loại hoạt động sinh học (ví dụ: ức chế, kích hoạt). Phân tích này sẽ tập trung vào việc xây dựng mô hình dự đoán hoạt tính của các hợp chất dựa trên cấu trúc hóa học của chúng.

Mô hình phân tích (Analysis Model):

- **Thu thập dữ liệu (Data Acquisition):** Lấy dữ liệu từ cơ sở dữ liệu ChEMBL 35.
- **Tiền xử lý dữ liệu (Data Preprocessing):**
    - Làm sạch dữ liệu: Loại bỏ các giá trị không hợp lệ, trùng lặp.
    - Chuẩn hóa dữ liệu: Đảm bảo tính nhất quán của dữ liệu.
    - Tính toán các thuộc tính cấu trúc (Molecular Descriptors): Sử dụng RDKit để tính toán các thuộc tính hóa học từ SMILES (Simplified Molecular Input Line Entry System).
- **Lựa chọn đặc trưng (Feature Selection):** Chọn các thuộc tính quan trọng nhất để xây dựng mô hình.
- **Xây dựng mô hình (Model Building):** Sử dụng các thuật toán học máy (ví dụ: Random Forest, Support Vector Machines) để xây dựng mô hình dự đoán.
- **Đánh giá mô hình (Model Evaluation):** Đánh giá hiệu suất của mô hình bằng các chỉ số phù hợp (ví dụ: RMSE, R^2).

2. Hướng dẫn song ngữ (Bilingual Instructions)

Tiếng Anh (English):

This analysis focuses on building a predictive model for compound activity based on their chemical structures. The process involves data acquisition from ChEMBL 35, preprocessing (cleaning, standardization), calculating molecular descriptors using RDKit, feature selection, model building using machine learning algorithms, and model evaluation.

Tiếng Việt (Vietnamese):

Phân tích này tập trung vào việc xây dựng mô hình dự đoán hoạt tính của các hợp chất dựa trên cấu trúc hóa học của chúng. Quá trình này bao gồm thu thập dữ liệu từ ChEMBL 35, tiền xử lý dữ liệu (làm sạch, chuẩn hóa), tính toán các thuộc tính phân tử bằng RDKit, lựa chọn đặc trưng, xây dựng mô hình bằng các thuật toán học máy, và đánh giá mô hình.

3. Code SQL và Python (SQL and Python Code)

SQL (để lấy dữ liệu từ ChEMBL 35):

```sql
-- Select 100 rows for demonstration purposes
SELECT DISTINCT
    act.molregno,
    act.standard_value,
    act.standard_units,
```

```
    act.standard_type,
    md.chembl_id,
    md.canonical_smiles
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'  -- You can change this to your desired activity type
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0 -- Exclude zero values
  AND act.standard_value ~ '^[0-9\.]+$' -- Ensure standard_value is numeric
LIMIT 100;
```

Giải thích (Explanation):

- `SELECT DISTINCT`: Chọn các hàng duy nhất.
- `act.molregno`: ID phân tử.
- `act.standard_value`: Giá trị hoạt tính.
- `act.standard_units`: Đơn vị hoạt tính.
- `act.standard_type`: Loại hoạt tính (ví dụ: IC50).
- `md.chembl_id`: ChEMBL ID.
- `md.canonical_smiles`: SMILES string (biểu diễn cấu trúc hóa học).
- `FROM activities act JOIN molecule_dictionary md ON act.molregno = md.molregno`: Kết nối hai bảng `activities` và `molecule_dictionary` dựa trên `molregno`.
- `WHERE act.standard_type = 'IC50'`: Lọc theo loại hoạt tính.
- `AND act.standard_units = 'nM'`: Lọc theo đơn vị hoạt tính.
- `AND act.standard_value IS NOT NULL`: Loại bỏ các giá trị NULL.
- `AND act.standard_value > 0`: Loại bỏ các giá trị bằng 0.
- `AND act.standard_value ~ '^[0-9\.]+$'`: **Sửa lỗi (Fixing the Error)**: Đảm bảo rằng `standard_value` là một số. Biểu thức chính quy `^[0-9\.]+$` kiểm tra xem chuỗi chỉ chứa các chữ số và dấu chấm.
- `LIMIT 100`: Giới hạn kết quả ở 100 hàng.

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

```
import os
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.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# Define base path
base_path = '.'  # Adjust as needed

# File path to the CSV file
csv_file_path = os.path.join(base_path, 'data', 'chembl_data.csv') # Replace
'chembl_data.csv' with your actual filename

# Load data from CSV
try:
    df = pd.read_csv(csv_file_path)
    print("Data loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
```

```
    exit()

# Data Cleaning and Preprocessing
df = df.dropna(subset=['canonical_smiles', 'standard_value']) # Drop rows with missing
SMILES or standard_value

# Convert standard_value to numeric and filter out non-numeric values
df = df[pd.to_numeric(df['standard_value'], errors='coerce').notna()]
df['standard_value'] = pd.to_numeric(df['standard_value'])

# Function to calculate molecular descriptors using RDKit
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    for name, func in Descriptors.descList:
        try:
            descriptors[name] = func(mol)
        except:
            descriptors[name] = np.nan  # Handle potential errors during descriptor
calculation
    return pd.Series(descriptors)

# Apply the descriptor calculation to each SMILES string
descriptors_df = df['canonical_smiles'].apply(calculate_descriptors)

# Merge descriptors with the main dataframe
df = pd.concat([df, descriptors_df], axis=1)

# Remove rows with NaN values in descriptors
df = df.dropna(axis=0)

# Feature Selection (Example: Select top 10 descriptors)
# You can use more sophisticated feature selection techniques here
X = df.drop(columns=['molregno', 'chembl_id', 'canonical_smiles', 'standard_units',
'standard_type'])
y = df['standard_value']

# Scale the features
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)

# Model Building (Random Forest Regressor)
model = RandomForestRegressor(n_estimators=100, random_state=42)  # You can adjust
hyperparameters
model.fit(X_train, y_train)

# Model Evaluation
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}")
```

Giải thích (Explanation):

- **Import Libraries:** Import các thư viện cần thiết.
- **Load Data:** Tải dữ liệu từ file CSV.
- **Data Cleaning:** Loại bỏ các hàng có giá trị `NaN` trong cột `canonical_smiles` và `standard_value`.
- **Descriptor Calculation:** Sử dụng RDKit để tính toán các thuộc tính phân tử từ SMILES string.
- **Feature Selection:** Chọn các thuộc tính quan trọng nhất (ví dụ: sử dụng SelectKBest).
- **Data Scaling:** Chuẩn hóa dữ liệu bằng StandardScaler.
- **Model Building:** Xây dựng mô hình Random Forest Regressor.
- **Model Evaluation:** Đánh giá mô hình bằng RMSE và R^2.

4. Ví dụ Code SQL và Python mẫu (Example SQL and Python Code)

Ví dụ 1: Lọc theo mục tiêu cụ thể (Filtering by Specific Target)

SQL:

```sql
SELECT DISTINCT
    act.molregno,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    md.chembl_id,
    md.canonical_smiles
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
JOIN target_dictionary td ON act.tid = td.tid
WHERE td.chembl_id = 'CHEMBL205'  -- Example target: 'CHEMBL205' (e.g., Dopamine D4
receptor)
  AND act.standard_type = 'Ki'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 0
  AND act.standard_value ~ '^[0-9\.]+$'
LIMIT 100;
```

Python:

```python
# After loading data, filter based on a specific range of activity values
df_filtered = df[(df['standard_value'] >= 100) & (df['standard_value'] <= 1000)]
print(f"Number of compounds after filtering: {len(df_filtered)}")
```

Ví dụ 2: Tính toán thuộc tính LogP (Calculating LogP)

Python:

```python
from rdkit.Chem import AllChem
from rdkit.Chem import Crippen

def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    return Crippen.MolLogP(mol)

df['logp'] = df['canonical_smiles'].apply(calculate_logp)
df = df.dropna(subset=['logp'])
print(df[['canonical_smiles', 'logp']].head())
```

Ví dụ 3: Sử dụng Support Vector Regression (Using Support Vector Regression)

Python:

```python
from sklearn.svm import SVR

# Model Building (Support Vector Regression)
model = SVR(kernel='rbf')  # You can adjust hyperparameters
model.fit(X_train, y_train)

# Model Evaluation
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 (SVR): {rmse}")
print(f"R^2 (SVR): {r2}")
```

Ví dụ 4: Lưu kết quả vào file CSV (Saving Results to CSV)

Python:

```python
# After model evaluation, save the predictions to a CSV file
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
results.to_csv(os.path.join(base_path, 'data', 'predictions.csv'), index=False)
```

Ví dụ 5: Lọc theo hoạt tính (Filtering by Activity values)

SQL:

```sql
SELECT DISTINCT
    act.molregno,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    md.chembl_id,
    md.canonical_smiles
FROM activities act
JOIN molecule_dictionary md ON act.molregno = md.molregno
WHERE act.standard_type = 'IC50'  -- You can change this to your desired activity type
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND act.standard_value > 10 and  act.standard_value < 100 -- Exclude zero values
  AND act.standard_value ~ '^[0-9\.]+$' -- Ensure standard_value is numeric
LIMIT 100;
```

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