PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Đồ thị mạng lưới target-ligand bằng NetworkX

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 sử dụng RDKit. Với vai trò chuyên gia R&D dược học, tôi sẽ giúp bạn xây dựng quy trình phân tích hiệu quả, từ truy vấn SQL đến xử lý và mô hình hóa dữ liệu bằng Python.

Dưới đây là phân tích chi tiết và hướng dẫn song ngữ cho chủ đề "Topic_CheMBL_35_76", cùng với các ví dụ code SQL và Python mẫu, giúp bạn bắt đầu dự án một cách suôn sẻ:

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

- **Mục tiêu (Objective):**
    - Phân tích mối quan hệ giữa cấu trúc hóa học (chemical structure) và hoạt tính sinh học (biological activity) của các hợp chất trong cơ sở dữ liệu ChEMBL 35.
    - Xây dựng mô hình dự đoán hoạt tính dựa trên cấu trúc phân tử.
- **Dữ liệu (Data):**
    - Dữ liệu từ cơ sở dữ liệu ChEMBL 35, bao gồm thông tin về hợp chất (compounds), hoạt tính sinh học (bioactivities), mục tiêu (targets), và các thuộc tính liên quan.
    - Sử dụng RDKit để tính toán các descriptor phân tử (molecular descriptors) từ cấu trúc hóa học (SMILES).
- **Phương pháp (Methodology):**
    1. **Truy vấn và Trích xuất Dữ liệu (Data Query and Extraction):**
        - Sử dụng SQL để truy vấn dữ liệu liên quan từ cơ sở dữ liệu ChEMBL 35.
        - Lọc dữ liệu dựa trên các tiêu chí cụ thể (ví dụ: loại hoạt tính, mục tiêu, giá trị hoạt tính).
    2. **Tiền xử lý Dữ liệu (Data Preprocessing):**
        - Làm sạch và chuẩn hóa dữ liệu.
        - Xử lý các giá trị thiếu (missing values) và ngoại lệ (outliers).
    3. **Tính toán Descriptor Phân tử (Molecular Descriptor Calculation):**
        - Sử dụng RDKit để chuyển đổi cấu trúc SMILES thành các descriptor phân tử (ví dụ: trọng lượng phân tử, logP, số lượng liên kết, v.v.).
    4. **Phân tích Thống kê (Statistical Analysis):**
        - Thực hiện phân tích thống kê mô tả (descriptive statistics) để hiểu rõ hơn về dữ liệu.
        - Sử dụng các phương pháp thống kê (ví dụ: hồi quy tuyến tính, tương quan) để xác định mối quan hệ giữa descriptor và hoạt tính.
    5. **Mô hình hóa (Modeling):**
        - Xây dựng mô hình học máy (machine learning models) để dự đoán hoạt tính sinh học dựa trên các descriptor phân tử.
        - Sử dụng các thuật toán như Random Forest, Support Vector Machines (SVM), hoặc Neural Networks.
    6. **Đánh giá Mô hình (Model Evaluation):**
        - Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số phù hợp (ví dụ: R-squared, RMSE, AUC).
        - Sử dụng cross-validation để đảm bảo tính tổng quát của mô hình.

2. Hướng dẫn Song ngữ (Bilingual Instructions)

- **SQL:**
    - *Tiếng Anh:* Use SQL to query and extract relevant data from the ChEMBL 35 database. Filter data based on specific criteria (e.g., activity type, target, activity values).
    - *Tiếng Việt:* Sử dụng SQL để truy vấn và trích xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL 35. Lọc dữ liệu dựa trên các tiêu chí cụ thể (ví dụ: loại hoạt tính, mục tiêu, giá trị hoạt tính).
- **Python:**
    - *Tiếng Anh:* Use RDKit to calculate molecular descriptors from SMILES structures. Preprocess data to handle missing values and outliers. Build and evaluate machine learning models to predict biological activity.
    - *Tiếng Việt:* Sử dụng RDKit để tính toán các descriptor phân tử từ cấu trúc SMILES. Tiền xử lý dữ liệu để xử lý các giá trị thiếu và ngoại lệ. Xây dựng và đánh giá các mô hình học máy để dự đoán hoạt tính sinh học.

3. Ví dụ Code (Code Examples)

SQL (English & Vietnamese)

```sql
-- English: SQL query to extract 100 compounds with IC50 values for a specific target (e.g., CHEMBL244)
-- Vietnamese: Truy vấn SQL để trích xuất 100 hợp chất có giá trị IC50 cho một mục tiêu cụ thể (ví dụ: CHEMBL244)
SELECT DISTINCT ON (cmpd.chembl_id)
    cmpd.chembl_id,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    mol.smiles
FROM
    compound_structures AS mol
    JOIN activities AS act ON mol.molregno = act.molregno
    JOIN assays AS ass ON act.assay_id = ass.assay_id
    JOIN target_dictionary AS td ON ass.tid = td.tid
    JOIN molecule_dictionary AS cmpd ON mol.molregno = cmpd.molregno
WHERE
    td.chembl_id = 'CHEMBL244' -- Replace with your target of interest / Thay thế bằng mục tiêu bạn quan tâm
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND mol.smiles IS NOT NULL
    AND act.standard_value ~ '^[0-9\.]+$' -- Ensure standard_value is numeric / Đảm bảo standard_value là số
LIMIT 100;
```

Python (English & Vietnamese)

```python
# English: Python code to read the extracted data, calculate molecular descriptors using RDKit, and train a Random Forest model
# Vietnamese: Mã Python để đọc dữ liệu đã trích xuất, tính toán descriptor phân tử bằng RDKit và huấn luyện mô hình Random Forest

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

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import os

# Define base path
base_path = "." # Assuming the notebook is in the root directory
data_path = os.path.join(base_path, "data")

# Load data from CSV
try:
    df = pd.read_csv(os.path.join(data_path, "chembl_ic50_data.csv")) # Adjust
filename accordingly
except FileNotFoundError:
    print(f"Error: File not found at {os.path.join(data_path,
'chembl_ic50_data.csv')}")
    exit()

# Function to calculate molecular descriptors
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] = None  # Handle potential errors during descriptor
calculation
    return descriptors

# Apply descriptor calculation
df['descriptors'] = df['smiles'].apply(calculate_descriptors)

# Handle errors and exclude rows with None descriptors
df = df.dropna(subset=['descriptors'])

# Convert descriptors to a DataFrame
descriptors_df = pd.DataFrame(df['descriptors'].tolist())

# Check for completely empty columns
empty_cols = descriptors_df.columns[descriptors_df.isnull().all()]
descriptors_df.drop(empty_cols, axis=1, inplace=True) #Remove empty columns

# Combine descriptors with activity data
X = descriptors_df
y = df['standard_value'].astype(float)  # Convert to numeric, handle potential
conversion errors

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

# Handle missing values using imputation (replace NaN with mean)
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
```

```
X_test = imputer.transform(X_test)

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

# Make predictions
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}")
```

Giải thích lỗi và cách khắc phục (Error Explanation and Fixes)

- **Lỗi SQL (SQL Error):**
    - *English:* `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`
    - *Vietnamese:* `Lỗi: không tồn tại toán tử: numeric ~ unknown, DÒNG 12: VÀ act.standard_value ~ '^[0-9\.]+$'`
    - *Explanation:* The `~` operator is used for regular expression matching, but PostgreSQL might not be able to implicitly convert the `standard_value` column to a text type for the regex comparison.
    - *Fix:* Cast the `standard_value` column to `TEXT` before applying the regex:

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

- **Lỗi Python (Python Error):**
    - *English:* "Old scikit-learn version does not support `squared=False` in `mean_squared_error`"
    - *Vietnamese:* "Phiên bản scikit-learn cũ không hỗ trợ tham số `squared=False` trong hàm `mean_squared_error`"
    - *Explanation:* The `squared=False` parameter was introduced in a later version of scikit-learn.
    - *Fix:* Remove the `squared=False` argument. The default behavior is to return the mean squared error (MSE). If you need the Root Mean Squared Error (RMSE), take the square root of the MSE:

      ```
      mse = mean_squared_error(y_test, y_pred)
      rmse = np.sqrt(mse)
      print(f"Root Mean Squared Error: {rmse}")
      ```

4. Thêm 4 ví dụ code .sql và python mẫu (4 More Code Examples)

SQL Example 2: Extracting data for a specific protein family

```
-- English: Extracting data for a specific protein family (e.g., Kinases)
-- Vietnamese: Trích xuất dữ liệu cho một họ protein cụ thể (ví dụ: Kinases)
SELECT DISTINCT ON (cmpd.chembl_id)
    cmpd.chembl_id,
```

```
    act.standard_value,
    act.standard_units,
    act.standard_type,
    mol.smiles
FROM
    compound_structures AS mol
    JOIN activities AS act ON mol.molregno = act.molregno
    JOIN assays AS ass ON act.assay_id = ass.assay_id
    JOIN target_dictionary AS td ON ass.tid = td.tid
    JOIN molecule_dictionary AS cmpd ON mol.molregno = cmpd.molregno
WHERE
    td.target_type = 'PROTEIN FAMILY'
    AND td.pref_name = 'Kinases'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND mol.smiles IS NOT NULL
    AND act.standard_value::TEXT ~ '^[0-9\.]+$'
LIMIT 100;
```

Python Example 2: Feature Importance from Random Forest

```
# English: Python code to display feature importance from the Random Forest model
# Vietnamese: Mã Python để hiển thị tầm quan trọng của các đặc trưng từ mô hình Random Forest

import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import os

# Define base path
base_path = "." # Assuming the notebook is in the root directory
data_path = os.path.join(base_path, "data")

# Load data from CSV
try:
    df = pd.read_csv(os.path.join(data_path, "chembl_ic50_data.csv")) # Adjust filename accordingly
except FileNotFoundError:
    print(f"Error: File not found at {os.path.join(data_path, 'chembl_ic50_data.csv')}")
    exit()

# Function to calculate molecular descriptors
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] = None  # Handle potential errors during descriptor
```

```
calculation
    return descriptors

# Apply descriptor calculation
df['descriptors'] = df['smiles'].apply(calculate_descriptors)

# Handle errors and exclude rows with None descriptors
df = df.dropna(subset=['descriptors'])

# Convert descriptors to a DataFrame
descriptors_df = pd.DataFrame(df['descriptors'].tolist())

# Check for completely empty columns
empty_cols = descriptors_df.columns[descriptors_df.isnull().all()]
descriptors_df.drop(empty_cols, axis=1, inplace=True) #Remove empty columns

# Combine descriptors with activity data
X = descriptors_df
y = df['standard_value'].astype(float)  # Convert to numeric, handle potential
conversion errors

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

# Handle missing values using imputation (replace NaN with mean)
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

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

# Get feature importances
importances = model.feature_importances_

# Create a DataFrame to display feature importances
feature_importances = pd.DataFrame({'Feature': descriptors_df.columns, 'Importance':
importances})
feature_importances = feature_importances.sort_values('Importance', ascending=False)

print(feature_importances)
```

SQL Example 3: Extracting data based on Lipinski's Rule of Five

```
-- English: Extracting compounds that satisfy Lipinski's Rule of Five
-- Vietnamese: Trích xuất các hợp chất thỏa mãn Quy tắc 5 của Lipinski
SELECT DISTINCT ON (cmpd.chembl_id)
    cmpd.chembl_id,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    mol.smiles
FROM
    compound_structures AS mol
    JOIN activities AS act ON mol.molregno = act.molregno
    JOIN molecule_dictionary AS cmpd ON mol.molregno = cmpd.molregno
WHERE
```

```
    cmpd.mw_freebase <= 500  -- Molecular weight <= 500
    AND cmpd.alogp <= 5      -- LogP <= 5
    AND cmpd.psa <= 140      -- Polar Surface Area <= 140
    AND cmpd.hba <= 10       -- Hydrogen Bond Acceptors <= 10
    AND cmpd.hbd <= 5        -- Hydrogen Bond Donors <= 5
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND mol.smiles IS NOT NULL
    AND act.standard_value::TEXT ~ '^[0-9\.]+$'
LIMIT 100;
```

Python Example 3: Using different Machine learning Model (SVM)

```
# English: Python code to train a Support Vector Machine (SVM) model
# Vietnamese: Mã Python để huấn luyện mô hình Support Vector Machine (SVM)

import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import os

# Define base path
base_path = "." # Assuming the notebook is in the root directory
data_path = os.path.join(base_path, "data")

# Load data from CSV
try:
    df = pd.read_csv(os.path.join(data_path, "chembl_ic50_data.csv")) # Adjust
filename accordingly
except FileNotFoundError:
    print(f"Error: File not found at {os.path.join(data_path,
'chembl_ic50_data.csv')}")
    exit()

# Function to calculate molecular descriptors
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] = None  # Handle potential errors during descriptor
calculation
    return descriptors

# Apply descriptor calculation
df['descriptors'] = df['smiles'].apply(calculate_descriptors)

# Handle errors and exclude rows with None descriptors
df = df.dropna(subset=['descriptors'])
```

```python
# Convert descriptors to a DataFrame
descriptors_df = pd.DataFrame(df['descriptors'].tolist())

# Check for completely empty columns
empty_cols = descriptors_df.columns[descriptors_df.isnull().all()]
descriptors_df.drop(empty_cols, axis=1, inplace=True) #Remove empty columns

# Combine descriptors with activity data
X = descriptors_df
y = df['standard_value'].astype(float)  # Convert to numeric, handle potential
conversion errors

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

# Handle missing values using imputation (replace NaN with mean)
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Train an SVM model
model = SVR(kernel='rbf')  # You can experiment with different kernels (e.g.,
'linear', 'poly')
model.fit(X_train, y_train)

# Make predictions
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}")
```

SQL Example 4: Using a specific assay type

```sql
-- English: Extracting data for a specific assay type (e.g., 'Binding')
-- Vietnamese: Trích xuất dữ liệu cho một loại assay cụ thể (ví dụ: 'Binding')
SELECT DISTINCT ON (cmpd.chembl_id)
    cmpd.chembl_id,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    mol.smiles
FROM
    compound_structures AS mol
    JOIN activities AS act ON mol.molregno = act.molregno
    JOIN assays AS ass ON act.assay_id = ass.assay_id
    JOIN target_dictionary AS td ON ass.tid = td.tid
    JOIN molecule_dictionary AS cmpd ON mol.molregno = cmpd.molregno
WHERE
    ass.assay_type = 'B'
    AND act.standard_type = 'Ki'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND mol.smiles IS NOT NULL
```

```
    AND act.standard_value::TEXT ~ '^[0-9\.]+$'
LIMIT 100;
```

Python Example 4: Scaling Descriptor with StandardScaler

```
# English: Python code StandardScaler scaling for descriptor data
# Vietnamese: Mã Python chuẩn hóa StandardScaler cho dữ liệu descriptor

import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import os

# Define base path
base_path = "." # Assuming the notebook is in the root directory
data_path = os.path.join(base_path, "data")

# Load data from CSV
try:
    df = pd.read_csv(os.path.join(data_path, "chembl_ic50_data.csv")) # Adjust
filename accordingly
except FileNotFoundError:
    print(f"Error: File not found at {os.path.join(data_path,
'chembl_ic50_data.csv')}")
    exit()

# Function to calculate molecular descriptors
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] = None  # Handle potential errors during descriptor
calculation
    return descriptors

# Apply descriptor calculation
df['descriptors'] = df['smiles'].apply(calculate_descriptors)

# Handle errors and exclude rows with None descriptors
df = df.dropna(subset=['descriptors'])

# Convert descriptors to a DataFrame
descriptors_df = pd.DataFrame(df['descriptors'].tolist())

# Check for completely empty columns
empty_cols = descriptors_df.columns[descriptors_df.isnull().all()]
descriptors_df.drop(empty_cols, axis=1, inplace=True) #Remove empty columns

# Combine descriptors with activity data
X = descriptors_df
y = df['standard_value'].astype(float)  # Convert to numeric, handle potential
```

```
conversion errors

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

# Handle missing values using imputation (replace NaN with mean)
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Standardize the feature with StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


# Train an SVM model
model = SVR(kernel='rbf')  # You can experiment with different kernels (e.g.,
'linear', 'poly')
model.fit(X_train_scaled, y_train)

# Make predictions
y_pred = model.predict(X_test_scaled)

# 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}")
```

Lưu ý quan trọng (Important Notes):

- **Đường dẫn (Paths):** Đảm bảo rằng đường dẫn đến file CSV của bạn chính xác. Sử dụng `os.path.join(base_path, "data", "your_file.csv")` để đảm bảo tính di động của code.
- **Xử lý lỗi (Error Handling):** Thêm các khối `try...except` để xử lý các lỗi có thể xảy ra trong quá trình tính toán descriptor hoặc đọc file.
- **Phiên bản thư viện (Library Versions):** Đảm bảo rằng bạn đang sử dụng các phiên bản thư viện tương thích. Nếu bạn gặp lỗi liên quan đến `squared=False`, hãy nâng cấp scikit-learn hoặc loại bỏ tham số đó.
- **Lọc dữ liệu (Data Filtering):** Lọc dữ liệu cẩn thận để đảm bảo rằng bạn đang làm việc với dữ liệu chất lượng cao. Loại bỏ các hợp chất có SMILES không hợp lệ hoặc các giá trị hoạt tính không đáng tin cậy.
- **Tuỳ chỉnh mô hình (Model Tuning):** Thử nghiệm với các thuật toán học máy khác nhau và các tham số của chúng để tìm ra mô hình tốt nhất cho dữ liệu của bạn. Sử dụng cross-validation để đánh giá hiệu suất của mô hình một cách khách quan.
- **Lưu trữ kết quả (Result saving):** Lưu trữ các mô hình, descriptor đã tính toán để có thể tái sử dụng.

Chúc bạn thành công với dự án của mình! Hãy cho tôi biết nếu bạn có bất kỳ câu hỏi nào khác.