PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Mô hình học sâu để dự đoán hoạt tính

Tuyệt vời! Với vai trò là một chuyên gia R&D dược học, tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit này. Dựa trên thông tin bạn cung cấp, tôi sẽ giúp bạn phân tích, hướng dẫn, cung cấp code mẫu SQL và Python để bạn có thể bắt đầu dự án một cách hiệu quả.

1. Phân tích mô hình phân tích (Analysis of the analysis model):

Chúng ta sẽ tập trung vào việc trích xuất và tiền xử lý dữ liệu từ cơ sở dữ liệu ChEMBL 35, sau đó sử dụng RDKit để tính toán các đặc trưng phân tử (molecular descriptors) và cuối cùng xây dựng mô hình dự đoán hoạt tính sinh học. Mô hình phân tích sẽ bao gồm các bước sau:

- **Trích xuất dữ liệu (Data Extraction):** Sử dụng SQL để lấy thông tin về các hợp chất và hoạt tính của chúng 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 hoặc thiếu, và chuẩn hóa dữ liệu (ví dụ: chuyển đổi IC50 thành pIC50).
- **Tính toán đặc trưng phân tử (Molecular Descriptor Calculation):** Sử dụng RDKit để tính toán các đặc trưng phân tử như trọng lượng phân tử, hệ số phân vùng octanol-nước (LogP), diện tích bề mặt phân cực (TPSA), và các đặc trưng hình thái khác.
- **Phân tích dữ liệu (Data Analysis):** Sử dụng các kỹ thuật thống kê và học máy để phân tích dữ liệu và xây dựng mô hình dự đoán hoạt tính sinh học.
- **Đánh giá mô hình (Model Evaluation):** Đánh giá hiệu suất của mô hình bằng cách sử dụng các chỉ số như R-squared, RMSE, MAE, và AUC.

2. Hướng dẫn song ngữ (Bilingual Instructions):

English:

This project aims to analyze ChEMBL 35 data using RDKit to support drug discovery and development. We will extract data from the ChEMBL 35 database, preprocess it, calculate molecular descriptors using RDKit, and build a predictive model for biological activity.

Tiếng Việt:

Dự án này nhằm mục đích phân tích dữ liệu ChEMBL 35 bằng RDKit để hỗ trợ nghiên cứu và phát triển thuốc. Chúng ta sẽ trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL 35, tiền xử lý dữ liệu, tính toán các đặc trưng phân tử bằng RDKit và xây dựng mô hình dự đoán hoạt tính sinh học.

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

3.1 SQL Code (for data extraction):

```sql
-- English: Extracting 100 compounds and their activity data from ChEMBL 35
-- Tiếng Việt: Trích xuất 100 hợp chất và dữ liệu hoạt tính của chúng từ ChEMBL 35
SELECT DISTINCT
    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
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
LIMIT 100;
```

Lưu ý về lỗi SQL:

Lỗi `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'` xảy ra do bạn đang cố gắng sử dụng toán tử `~` (regular expression match) trên một cột kiểu số (numeric). Để khắc phục, bạn có thể sử dụng hàm `CAST` để chuyển đổi cột `standard_value` thành kiểu text trước khi so sánh:

```
-- English: Corrected SQL query to handle numeric data type
-- Tiếng Việt: Sửa truy vấn SQL để xử lý kiểu dữ liệu số
SELECT DISTINCT
    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
WHERE act.standard_type = 'IC50'
  AND act.standard_units = 'nM'
  AND act.standard_value IS NOT NULL
  AND CAST(act.standard_value AS TEXT) SIMILAR TO '[0-9.]+'
LIMIT 100;
```

3.2 Python Code (for data processing and analysis):

```
# English: Python code for data processing and analysis
# Tiếng Việt: Mã Python để xử lý và phân tích dữ liệu

import pandas as pd
import numpy as np
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, r2_score
import os

# Define base path
base_path = "."  # Assuming the notebook is in the root of the project

# Construct the data path
data_path = os.path.join(base_path, "data", "chembl_data.csv") # Assuming your data
file is named chembl_data.csv

# Load data from CSV file
try:
    df = pd.read_csv(data_path)
    print("Data loaded successfully!")
except FileNotFoundError:
```

```
    print(f"Error: File not found at {data_path}. Please make sure the file exists.")
    exit()
except Exception as e:
    print(f"Error loading data: {e}")
    exit()


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

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

# Handle missing descriptors
df = df.dropna(subset=['descriptors'])

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

# Convert IC50 to pIC50
df['pIC50'] = -np.log10(df['standard_value'].astype(float) / 1e9)

# Select features and target
features = ['MolWt', 'LogP', 'HBD', 'HBA', 'TPSA']
target = 'pIC50'

# Prepare data for modeling
X = df[features]
y = df[target]

# 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)

# Create and train a linear regression model
model = LinearRegression()
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}")
```

Lưu ý về lỗi Python:

Lỗi "phiên bản scikit-learn cũ không hỗ trợ tham số squared=False trong hàm mean_squared_error" không còn là vấn đề vì bạn không sử dụng tham số `squared=False` trong code.

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

Dưới đây là 5 ví dụ về các truy vấn SQL và code Python khác nhau mà bạn có thể sử dụng trong dự án của mình:

Ví dụ 1: Tìm kiếm các hợp chất có hoạt tính cao nhất (Finding the most active compounds):

SQL:

```sql
-- English: Find the top 10 most active compounds based on IC50 values
-- Tiếng Việt: Tìm 10 hợp chất có hoạt tính cao nhất dựa trên giá trị IC50
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_units = 'nM'
ORDER BY act.standard_value ASC
LIMIT 10;
```

Python:

```python
# English: Calculate and display the correlation matrix of the descriptors
# Tiếng Việt: Tính toán và hiển thị ma trận tương quan của các descriptors
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix
correlation_matrix = df[features].corr()

# Plot the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix of Molecular Descriptors")
plt.show()
```

Ví dụ 2: Tính toán phân bố của LogP (Calculating LogP distribution):

SQL:

```sql
-- English: Calculate the average LogP value for compounds in ChEMBL 35
-- Tiếng Việt: Tính giá trị LogP trung bình cho các hợp chất trong ChEMBL 35
-- This requires you to have LogP values stored in the database.
-- Điều này yêu cầu bạn phải có giá trị LogP được lưu trữ trong cơ sở dữ liệu.

-- Assuming you have a table with LogP values, adapt this query accordingly.
-- Giả sử bạn có một bảng với giá trị LogP, hãy điều chỉnh truy vấn này cho phù hợp.
-- Example:
-- Ví dụ:
-- SELECT AVG(logp_value) FROM compounds;
```

Python:

```python
# English: Plot the distribution of pIC50 values
# Tiếng Việt: Vẽ biểu đồ phân bố giá trị pIC50
plt.figure(figsize=(8, 6))
sns.histplot(df['pIC50'], kde=True)
plt.title("Distribution of pIC50 Values")
plt.xlabel("pIC50")
```

```
plt.ylabel("Frequency")
plt.show()
```

Ví dụ 3: Lọc các hợp chất theo trọng lượng phân tử (Filtering compounds by molecular weight):

SQL:

```
-- English: Find compounds with molecular weight between 200 and 400
-- Tiếng Việt: Tìm các hợp chất có trọng lượng phân tử từ 200 đến 400
-- This requires you to have molecular weight values stored in the database.
-- Điều này yêu cầu bạn phải có giá trị trọng lượng phân tử được lưu trữ trong cơ sở dữ liệu.

-- Assuming you have a table with molecular weight values, adapt this query accordingly.
-- Giả sử bạn có một bảng với giá trị trọng lượng phân tử, hãy điều chỉnh truy vấn này cho phù hợp.
-- Example:
-- Ví dụ:
-- SELECT molregno, canonical_smiles FROM compounds WHERE mol_weight BETWEEN 200 AND 400;
```

Python:

```
# English: Train a Random Forest Regressor model
# Tiếng Việt: Huấn luyện mô hình Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

# Create and train a Random Forest Regressor model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions
y_pred_rf = rf_model.predict(X_test)

# Evaluate the model
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print(f"Random Forest Mean Squared Error: {mse_rf}")
print(f"Random Forest R-squared: {r2_rf}")
```

Ví dụ 4: Tìm kiếm các hợp chất có một khung cấu trúc nhất định (Searching for compounds with a specific scaffold):

SQL:

```
-- English: Find compounds containing a specific substructure (e.g., benzene ring)
-- Tiếng Việt: Tìm các hợp chất chứa một cấu trúc con cụ thể (ví dụ: vòng benzen)
-- This requires you to have a way to search for substructures in the database.
-- Điều này yêu cầu bạn phải có một cách để tìm kiếm các cấu trúc con trong cơ sở dữ liệu.

-- Assuming you have a function or table for substructure search, adapt this query accordingly.
-- Giả sử bạn có một hàm hoặc bảng để tìm kiếm cấu trúc con, hãy điều chỉnh truy vấn này cho phù hợp.
-- Example:
-- Ví dụ:
```

```
-- SELECT molregno, canonical_smiles FROM compounds WHERE
contains_substructure(canonical_smiles, 'c1ccccc1');
```

Python:

```
# English: Calculate and display feature importance from the Random Forest model
# Tiếng Việt: Tính toán và hiển thị tầm quan trọng của các đặc trưng từ mô hình Random
Forest
feature_importances = rf_model.feature_importances_

# Create a DataFrame to store feature importances
feature_importance_df = pd.DataFrame({'Feature': features, 'Importance':
feature_importances})

# Sort the DataFrame by importance
feature_importance_df = feature_importance_df.sort_values(by='Importance',
ascending=False)

# Print the feature importances
print("Feature Importances:")
print(feature_importance_df)

# Plot the feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title("Feature Importances in Random Forest Model")
plt.show()
```

Ví dụ 5: Tiền xử lý dữ liệu để chuẩn bị cho mô hình hóa (Data preprocessing for modeling):

SQL:

```
-- English: Extract data and convert IC50 to pIC50 within the SQL query
-- Tiếng Việt: Trích xuất dữ liệu và chuyển đổi IC50 thành pIC50 trong truy vấn SQL
SELECT
    md.molregno,
    cs.canonical_smiles,
    act.standard_value,
    -LOG(act.standard_value * 1e-9) AS pIC50  -- Convert IC50 (nM) to pIC50
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 IS NOT NULL
LIMIT 100;
```

Python:

```
# English: Standardize features using StandardScaler
# Tiếng Việt: Chuẩn hóa các đặc trưng bằng StandardScaler
from sklearn.preprocessing import StandardScaler

# Create a StandardScaler object
scaler = StandardScaler()

# Fit the scaler on the training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)

# Now use the scaled data for modeling
model = LinearRegression()
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:

- Hãy nhớ điều chỉnh các truy vấn SQL và code Python cho phù hợp với cấu trúc và nội dung cụ thể của cơ sở dữ liệu ChEMBL 35 của bạn.
- Đảm bảo rằng bạn đã cài đặt tất cả các thư viện Python cần thiết (ví dụ: pandas, RDKit, scikit-learn). Bạn có thể sử dụng pip để cài đặt chúng: `pip install pandas rdkit scikit-learn seaborn matplotlib`.
- Khi làm việc với dữ liệu lớn, hãy cân nhắc sử dụng các kỹ thuật tối ưu hóa để tăng tốc độ xử lý.

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