PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Đánh giá overfitting trong mô hình QSAR

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 này. Dựa trên thông tin bạn cung cấp, tôi sẽ đưa ra phân tích, hướng dẫn chi tiết và các ví dụ code mẫu để bạn có thể triển khai một cách hiệu quả.

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

Dựa trên mã chủ đề "Topic_CheMBL_35_67", tôi giả định rằng chủ đề của bạn liên quan đến việc phân tích mối quan hệ giữa cấu trúc hóa học (từ ChEMBL 35) và hoạt tính sinh học của các hợp chất. Dưới đây là một mô hình phân tích tiềm năng:

- **Mục tiêu:** Dự đoán hoặc phân loại hoạt tính sinh học của các hợp chất dựa trên cấu trúc hóa học của chúng.
- **Dữ liệu:**
    - **Đầu vào:** Dữ liệu cấu trúc hóa học từ ChEMBL 35 (SMILES strings, fingerprints, descriptors).
    - **Đầu ra:** Dữ liệu hoạt tính sinh học từ ChEMBL 35 (IC50, Ki, EC50, etc.).
- **Phương pháp:**
    - **Chuẩn bị dữ liệu:**
        - Làm sạch và chuẩn hóa dữ liệu cấu trúc hóa học (sử dụng RDKit).
        - Tính toán các descriptors (tính chất hóa lý) từ cấu trúc hóa học (sử dụng RDKit).
        - Xử lý dữ liệu hoạt tính sinh học (chuyển đổi, chuẩn hóa).
    - **Lựa chọn mô hình:**
        - **Hồi quy:** Linear Regression, Random Forest Regression, Support Vector Regression (SVR).
        - **Phân loại:** Logistic Regression, Random Forest Classifier, Support Vector Machine (SVM).
    - **Đánh giá mô hình:**
        - Sử dụng các metric phù hợp (R-squared, RMSE, MAE cho hồi quy; Accuracy, Precision, Recall, F1-score cho phân loại).
        - Sử dụng cross-validation để đánh giá khả năng tổng quát hóa của mô hình.

2. Hướng dẫn Song ngữ (Bilingual Instructions)

2.1. Thiết lập Môi trường (Environment Setup)

- **Tiếng Việt:** Đảm bảo bạn đã cài đặt các thư viện cần thiết: RDKit, scikit-learn, pandas, numpy, psycopg2 (nếu cần).
- **English:** Ensure you have installed the necessary libraries: RDKit, scikit-learn, pandas, numpy, psycopg2 (if needed).

```
# Tiếng Việt: Cài đặt các thư viện Python
# English: Install Python libraries
pip install rdkit-pypi scikit-learn pandas numpy psycopg2
```

2.2. Kết nối đến Cơ sở Dữ liệu ChEMBL 35 (Connecting to ChEMBL 35 Database)

- **Tiếng Việt:** Sử dụng thông tin đăng nhập bạn đã cung cấp để kết nối đến cơ sở dữ liệu ChEMBL 35.

- **English:** Use the credentials you provided to connect to the ChEMBL 35 database.

2.3. Truy vấn Dữ liệu (Data Querying)

- **Tiếng Việt:** Sử dụng SQL để truy vấn dữ liệu từ cơ sở dữ liệu ChEMBL 35.
- **English:** Use SQL to query data from the ChEMBL 35 database.

2.4. Xử lý Dữ liệu (Data Processing)

- **Tiếng Việt:** Sử dụng pandas và RDKit để xử lý và chuẩn bị dữ liệu cho mô hình.
- **English:** Use pandas and RDKit to process and prepare the data for the model.

2.5. Huấn luyện Mô hình (Model Training)

- **Tiếng Việt:** Sử dụng scikit-learn để huấn luyện mô hình dự đoán hoặc phân loại.
- **English:** Use scikit-learn to train the prediction or classification model.

2.6. Đánh giá Mô hình (Model Evaluation)

- **Tiếng Việt:** Đánh giá hiệu suất của mô hình sử dụng các metric phù hợp.
- **English:** Evaluate the model's performance using appropriate metrics.

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

3.1. SQL Code

```sql
-- English: Query to retrieve compound structures and bioactivity data
-- Tiếng Việt: Truy vấn để lấy cấu trúc hợp chất và dữ liệu hoạt tính sinh học
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
WHERE
    act.standard_type = 'IC50'  -- Chỉ lấy dữ liệu IC50 (Only retrieve IC50 data)
    AND act.standard_value IS NOT NULL  -- Giá trị không được NULL (Value must not be
NULL)
    AND act.standard_value > 0  -- Giá trị phải dương (Value must be positive)
    AND act.standard_units = 'nM' -- Chỉ lấy đơn vị nM (Only retrieve nM units)
    AND cs.canonical_smiles IS NOT NULL
    AND act.standard_value ~ '^[0-9\.]+$' -- Giá trị chỉ chứa số và dấu chấm (Value
contains only numbers and dots)
LIMIT 100;  -- Giới hạn 100 dòng (Limit to 100 rows)
```

Lưu ý:

- Lỗi `ERROR: operator does not exist: numeric ~ unknown` xảy ra vì bạn đang cố gắng so sánh một cột kiểu số (numeric) với một chuỗi (unknown). Để khắc phục, bạn có thể chuyển đổi cột `standard_value` sang kiểu text trước khi so sánh:

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

3.2. Python Code

```python
import os
import pandas as pd
```

```python
import numpy as np
from rdkit import Chem
from rdkit.Chem import AllChem
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Đường dẫn cơ sở (Base path)
base_path = "."  # Thay đổi nếu cần (Change if necessary)

# Đường dẫn đến file CSV (Path to CSV file)
csv_file_path = os.path.join(base_path, "data", "chembl_ic50_data.csv") #Sửa tên
file .csv sau khi chạy file .sql

# Đọc dữ liệu từ file CSV (Read data from CSV file)
try:
    data = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}")
    exit()

# In ra thông tin về dữ liệu (Print data information)
print(data.head())

# Hàm tính toán fingerprint (Function to calculate fingerprint)
def calculate_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

# Áp dụng hàm tính toán fingerprint (Apply the fingerprint calculation function)
data['fingerprint'] = data['canonical_smiles'].apply(calculate_fingerprint)

# Loại bỏ các hàng có fingerprint là None (Remove rows with None fingerprint)
data = data.dropna(subset=['fingerprint'])

# Chuyển đổi IC50 sang pIC50 (Convert IC50 to pIC50)
data['pIC50'] = -np.log10(data['standard_value'] * 1e-9)  # Chuyển đổi nM sang M
(Convert nM to M)

# Chuẩn bị dữ liệu cho mô hình (Prepare data for the model)
X = np.vstack(data['fingerprint'].values)
y = data['pIC50'].values

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra (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)

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

# Dự đoán trên tập kiểm tra (Predict on the test set)
y_pred = model.predict(X_test)
```

```
# Đánh giá mô hình (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:

- Nếu bạn gặp lỗi `squared=False` trong hàm `mean_squared_error`, hãy đảm bảo rằng bạn đang sử dụng phiên bản scikit-learn mới nhất. Nếu không, bạn có thể bỏ qua tham số `squared=False` (mặc định là `True`, trả về MSE thay vì RMSE).
- Đảm bảo bạn đã thay đổi `csv_file_path` thành đường dẫn chính xác đến file CSV của bạn.

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

Dưới đây là 5 ví dụ khác nhau về cách truy vấn và phân tích dữ liệu ChEMBL 35:

Ví dụ 1: Tìm các hợp chất có hoạt tính cao nhất đối với một mục tiêu cụ thể (Find compounds with the highest activity against a specific target)

- **SQL:**

```
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.target_chembl_id = 'CHEMBL205'  -- Thay đổi target_chembl_id nếu cần
(Change target_chembl_id if needed)
ORDER BY
    act.standard_value ASC
LIMIT 10;
```

- **Python:**

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

def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        return Descriptors.MolLogP(mol)
    else:
        return None

# Giả sử bạn đã có DataFrame 'df' từ kết quả truy vấn SQL
# Assume you already have a DataFrame 'df' from the SQL query results

# Ví dụ: Tạo DataFrame từ dữ liệu mẫu (Example: Create DataFrame from sample data)
data = {'canonical_smiles': ['CCO', 'c1ccccc1', 'C[C@@H](O)c1ccccc1']}
df = pd.DataFrame(data)
```

```
df['logP'] = df['canonical_smiles'].apply(calculate_logp)
print(df.head())
```

Ví dụ 2: Thống kê số lượng hợp chất cho mỗi loại hoạt tính (Count the number of compounds for each activity type)

- **SQL:**

```
SELECT
    standard_type,
    COUNT(*)
FROM
    activities
GROUP BY
    standard_type
ORDER BY
    COUNT(*) DESC;
```

- **Python:**

```
# Giả sử bạn đã có DataFrame 'df' từ kết quả truy vấn SQL
# Assume you already have a DataFrame 'df' from the SQL query results

# Ví dụ: Tạo DataFrame từ dữ liệu mẫu (Example: Create DataFrame from sample data)
data = {'standard_type': ['IC50', 'Ki', 'IC50', 'EC50', 'Ki']}
df = pd.DataFrame(data)

activity_counts = df['standard_type'].value_counts()
print(activity_counts)
```

Ví dụ 3: Tính toán phân tử lượng trung bình của các hợp chất (Calculate the average molecular weight of compounds)

- **SQL:**

```
SELECT
    AVG(mol_weight)
FROM
    molecule_dictionary;
```

- **Python:**

```
def calculate_mw(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        return Descriptors.MolWt(mol)
    else:
        return None

# Giả sử bạn đã có DataFrame 'df' từ kết quả truy vấn SQL với cột 'canonical_smiles'
# Assume you already have a DataFrame 'df' from the SQL query results with a
'canonical_smiles' column

# Ví dụ: Tạo DataFrame từ dữ liệu mẫu (Example: Create DataFrame from sample data)
data = {'canonical_smiles': ['CCO', 'c1ccccc1', 'C[C@@H](O)c1ccccc1']}
df = pd.DataFrame(data)

df['mol_weight'] = df['canonical_smiles'].apply(calculate_mw)
average_mw = df['mol_weight'].mean()
print(f"Average Molecular Weight: {average_mw}")
```

Ví dụ 4: Lọc các hợp chất dựa trên khoảng phân tử lượng (Filter compounds based on molecular weight range)

- **SQL:**

```sql
SELECT
    md.molregno,
    cs.canonical_smiles,
    md.mol_weight
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
WHERE
    md.mol_weight BETWEEN 200 AND 400;
```

- **Python:**

```python
# Giả sử bạn đã có DataFrame 'df' từ kết quả truy vấn SQL với cột 'mol_weight'
# Assume you already have a DataFrame 'df' from the SQL query results with a
'mol_weight' column

# Ví dụ: Tạo DataFrame từ dữ liệu mẫu (Example: Create DataFrame from sample data)
data = {'mol_weight': [150, 250, 350, 450]}
df = pd.DataFrame(data)

filtered_df = df[(df['mol_weight'] >= 200) & (df['mol_weight'] <= 400)]
print(filtered_df)
```

Ví dụ 5: Tính toán Tanimoto similarity giữa các fingerprints (Calculate Tanimoto similarity between fingerprints)

- **SQL:** (Không thể thực hiện trực tiếp trong SQL, cần sử dụng Python)
- **Python:**

```python
from rdkit import DataStructs

def calculate_tanimoto(fp1, fp2):
    return DataStructs.TanimotoSimilarity(fp1, fp2)

# Giả sử bạn đã có hai fingerprints (Assume you have two fingerprints)
# Ví dụ:
mol1 = Chem.MolFromSmiles('CCO')
fp1 = AllChem.GetMorganFingerprintAsBitVect(mol1, 2, nBits=2048)

mol2 = Chem.MolFromSmiles('c1ccccc1')
fp2 = AllChem.GetMorganFingerprintAsBitVect(mol2, 2, nBits=2048)

tanimoto_similarity = calculate_tanimoto(fp1, fp2)
print(f"Tanimoto Similarity: {tanimoto_similarity}")
```

Tôi hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn bắt đầu dự án phân tích dữ liệu ChEMBL 35 của mình một cách suôn sẻ. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi! Chúc bạn thành công!