PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

Copyright 2025 | RnD Pharma Plus | www.nghiencuuthuoc.com | Zalo: +84888999311 |

🧩 Topic: Linking ChEMBL to FDA label data

Okay, I understand. You're working with Chembl 35 data using RDKit and Jupyter notebooks, following an AIMLops template, and need assistance with structuring your code, addressing errors, and creating examples. I will provide SQL and Python code snippets, focusing on retrieving a limited dataset (100 rows), handling the specified errors, and adhering to your file structure.

Let's break down the process, focusing on clear explanations in both English and Vietnamese, and adhering to the naming conventions and file structure you provided.

1. Analysis of the Analysis Model (Phân tích mô hình phân tích)

We're essentially aiming to:

- **Extract relevant data:** Fetch specific activity and molecule information from the ChEMBL 35 database using SQL. We'll focus on filtering for activities with numerical standard values to avoid errors during numerical analysis.
- **Process data with RDKit:** Use RDKit to generate molecular fingerprints (or other descriptors) from the SMILES strings retrieved from the database.
- **Explore and Model:** Perform basic exploratory data analysis (EDA) and potentially build a simple predictive model (e.g., a regression model to predict activity based on molecular descriptors). We'll address the `squared=False` error in `mean_squared_error` by either upgrading scikit-learn or using alternative metrics.
- **Document and Organize:** Keep all code organized according to your AIMLops template, using Jupyter notebooks and appropriate file naming conventions.

2. Code (Mã)

Here's the breakdown of the code, focusing on the key steps, with explanations in both English and Vietnamese.

SQL (SQL)

- **File:** `../data/chembl_35_data_100.csv` (This file will be created after running the SQL query and exporting the results)

```sql
-- Topic_CheMBL_35_84.sql

SELECT
    md.chembl_id,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.pchembl_value
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'  -- Example activity type
    AND act.standard_relation = '='
    AND act.standard_value IS NOT NULL
```

```
    AND act.standard_units = 'nM' -- Only consider data with standard units of 'nM'
    AND act.pchembl_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$' -- Ensure standard_value is numeric
ORDER BY
    act.pchembl_value DESC
LIMIT 100;
```

- **English Explanation:**
    - This SQL query retrieves data from the `activities`, `molecule_dictionary`, and `compound_structures` tables in the ChEMBL database.
    - It filters for IC50 activity data with numerical standard values (in nM) and a defined pChEMBL value.
    - The `act.standard_value::text ~ '^[0-9\.]+$'` clause checks if the `standard_value` can be cast to text and matches a numeric pattern (allowing for decimal points). This addresses the "operator does not exist" error by explicitly casting the numeric column to text for regex matching.
    - The results are ordered by pChEMBL value in descending order and limited to the top 100 rows. This limits the dataset size for efficient processing.
- **Vietnamese Explanation:**
    - Câu truy vấn SQL này lấy dữ liệu từ các bảng `activities`, `molecule_dictionary` và `compound_structures` trong cơ sở dữ liệu ChEMBL.
    - Nó lọc dữ liệu hoạt động IC50 với các giá trị chuẩn là số (trong nM) và giá trị pChEMBL được xác định.
    - Mệnh đề `act.standard_value::text ~ '^[0-9\.]+$'` kiểm tra xem `standard_value` có thể được chuyển đổi thành văn bản và khớp với một mẫu số (cho phép dấu thập phân) hay không. Điều này giải quyết lỗi "operator does not exist" bằng cách chuyển đổi rõ ràng cột số thành văn bản để khớp regex.
    - Kết quả được sắp xếp theo giá trị pChEMBL theo thứ tự giảm dần và giới hạn ở 100 hàng đầu. Điều này giới hạn kích thước tập dữ liệu để xử lý hiệu quả.

Python (Python)

- **File:** `notebook/Topic_CheMBL_35_84_1_data_processing.ipynb`

```python
# Topic_CheMBL_35_84_1_data_processing.ipynb

import pandas as pd
import os
from rdkit import Chem
from rdkit.Chem import AllChem
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

# Define paths based on AIMLops template
base_path = os.path.abspath(os.path.join(os.getcwd(), '..')) # Go up one level
data_path = os.path.join(base_path, 'data')
model_path = os.path.join(base_path, 'models') # Example:  If you need to save models

# Create directories if they don't exist
os.makedirs(data_path, exist_ok=True)
os.makedirs(model_path, exist_ok=True)


# Load the data
```

```
data_file = os.path.join(data_path, 'chembl_35_data_100.csv')
try:
    df = pd.read_csv(data_file)
    print(f"Data loaded successfully from {data_file}")
except FileNotFoundError:
    print(f"Error: File not found at {data_file}.  Make sure you ran the SQL query and
saved the CSV.")
    exit()

# Basic Data Exploration
print(df.head())
print(df.info())
print(df.describe())

# RDKit processing (generating Morgan fingerprints)
def generate_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048) #radius 2, 2048
bits
        return np.array(fp)
    else:
        return None

df['fingerprint'] = df['canonical_smiles'].apply(generate_fingerprint)
df = df.dropna(subset=['fingerprint'])  # Remove rows where fingerprint generation
failed

# Data Preprocessing for Modeling
X = np.stack(df['fingerprint'].values)
y = df['pchembl_value'].values

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

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

# Model Evaluation
y_pred = model.predict(X_test)

# Handle the squared=False issue.  Easiest is to upgrade scikit-learn
# If you can't upgrade, use another metric like mean absolute error (MAE)
try:
    mse = mean_squared_error(y_test, y_pred) #, squared=False) # Only available in
newer scikit-learn
    print(f"Mean Squared Error: {mse}")
except TypeError as e:
    print(f"Error: {e}.  This likely means your scikit-learn version is too old. Using
MAE instead.")
    from sklearn.metrics import mean_absolute_error
    mae = mean_absolute_error(y_test, y_pred)
    print(f"Mean Absolute Error: {mae}")


r2 = r2_score(y_test, y_pred)
print(f"R-squared: {r2}")
```

```
# Example of saving the model (optional)
# import joblib
# model_filename = os.path.join(model_path, 'linear_regression_model.pkl')
# joblib.dump(model, model_filename)
# print(f"Model saved to {model_filename}")
```

- **English Explanation:**
    - This Python code loads the data from the CSV file you created.
    - It uses RDKit to generate Morgan fingerprints from the SMILES strings. Fingerprints are numerical representations of the molecule's structure.
    - It splits the data into training and testing sets.
    - It trains a simple linear regression model to predict pChEMBL value based on the fingerprints.
    - It calculates and prints the Mean Squared Error (or Mean Absolute Error if you have an older version of scikit-learn) and R-squared value to evaluate the model's performance.
    - It includes a commented-out example of how to save the trained model using `joblib`. This is optional but good practice for reproducibility and deployment.
    - Crucially, it handles the potential `squared=False` error by checking for it and using Mean Absolute Error (MAE) as an alternative metric if needed.
    - The code uses `os.path.join` to construct file paths, adhering to your AIMLops directory structure.
    - `os.makedirs` ensures the 'data' and 'models' directories exist before attempting to write files.
- **Vietnamese Explanation:**
    - Đoạn mã Python này tải dữ liệu từ tệp CSV bạn đã tạo.
    - Nó sử dụng RDKit để tạo dấu vân tay Morgan từ chuỗi SMILES. Dấu vân tay là biểu diễn số của cấu trúc phân tử.
    - Nó chia dữ liệu thành tập huấn luyện và tập kiểm tra.
    - Nó huấn luyện một mô hình hồi quy tuyến tính đơn giản để dự đoán giá trị pChEMBL dựa trên dấu vân tay.
    - Nó tính toán và in Sai số bình phương trung bình (hoặc Sai số tuyệt đối trung bình nếu bạn có phiên bản scikit-learn cũ hơn) và giá trị R bình phương để đánh giá hiệu suất của mô hình.
    - Nó bao gồm một ví dụ được chú thích về cách lưu mô hình đã huấn luyện bằng `joblib`. Điều này là tùy chọn nhưng là một thực hành tốt để tái tạo và triển khai.
    - Quan trọng nhất, nó xử lý lỗi tiềm ẩn `squared=False` bằng cách kiểm tra nó và sử dụng Sai số tuyệt đối trung bình (MAE) làm chỉ số thay thế nếu cần.
    - Mã sử dụng `os.path.join` để xây dựng đường dẫn tệp, tuân thủ cấu trúc thư mục AIMLops của bạn.
    - `os.makedirs` đảm bảo các thư mục 'data' và 'models' tồn tại trước khi cố gắng ghi tệp.

3. Examples (Ví dụ)

Here are 5 examples showing how you might extend this core code:

Example 1: Different Fingerprint Type (Ví dụ 1: Loại Dấu Vân Tay Khác)

```
#Instead of Morgan fingerprints, use RDKit's topological torsion fingerprints
from rdkit.Chem import rdMolDescriptors
def generate_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
```

```
    if mol:
        fp = rdMolDescriptors.GetHashedTopologicalTorsion(mol, nBits=2048)
        return np.array(fp)
    else:
        return None
```

- **English:** This example shows how to switch from Morgan fingerprints to topological torsion fingerprints. Simply change the `generate_fingerprint` function to use `rdMolDescriptors.GetHashedTopologicalTorsion`.
- **Vietnamese:** Ví dụ này cho thấy cách chuyển từ dấu vân tay Morgan sang dấu vân tay xoắn topo. Chỉ cần thay đổi hàm `generate_fingerprint` để sử dụng `rdMolDescriptors.GetHashedTopologicalTorsion`.

Example 2: Using Different Molecular Descriptors (Ví dụ 2: Sử dụng các Mô tả Phân tử Khác)

```
from rdkit.Chem import Descriptors

def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return [Descriptors.MolWt(mol), Descriptors.MolLogP(mol),
Descriptors.TPSA(mol)]  # Example descriptors
    else:
        return None

df['descriptors'] = df['canonical_smiles'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors'])

# Preprocessing:  Expand the descriptor list into separate columns
df[['mol_wt', 'logp', 'tpsa']] = pd.DataFrame(df['descriptors'].tolist(),
index=df.index)

X = df[['mol_wt', 'logp', 'tpsa']].values  # Use these as features
```

- **English:** This demonstrates how to use other molecular descriptors (e.g., molecular weight, LogP, TPSA) instead of fingerprints. You'll need to calculate the descriptors using functions from `rdkit.Chem.Descriptors`. Then, you need to expand the descriptor list into individual columns for use as features.
- **Vietnamese:** Điều này minh họa cách sử dụng các mô tả phân tử khác (ví dụ: trọng lượng phân tử, LogP, TPSA) thay vì dấu vân tay. Bạn cần tính toán các mô tả bằng cách sử dụng các hàm từ `rdkit.Chem.Descriptors`. Sau đó, bạn cần mở rộng danh sách mô tả thành các cột riêng lẻ để sử dụng làm tính năng.

Example 3: Different Regression Model (Ví dụ 3: Mô hình Hồi quy Khác)

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=100, random_state=42)  # Example parameters
model.fit(X_train, y_train)
```

- **English:** This shows how to use a Random Forest Regressor instead of Linear Regression. Simply import the desired model from `sklearn.ensemble` and instantiate it. You might need to adjust the model's parameters for optimal performance.
- **Vietnamese:** Điều này cho thấy cách sử dụng Random Forest Regressor thay vì Hồi quy tuyến tính. Chỉ cần nhập mô hình mong muốn từ `sklearn.ensemble` và khởi tạo nó. Bạn có thể cần điều chỉnh các tham số của mô hình để có hiệu suất tối ưu.

Example 4: Cross-Validation (Ví dụ 4: Xác thực chéo)

```
from sklearn.model_selection import cross_val_score

# Perform cross-validation
scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')  # Example: 5-fold cross-validation
rmse_scores = np.sqrt(-scores)  # Convert to RMSE

print("Cross-validation RMSE scores:", rmse_scores)
print("Mean RMSE:", rmse_scores.mean())
print("Standard deviation of RMSE:", rmse_scores.std())
```

- **English:** This adds cross-validation to your model evaluation. Cross-validation provides a more robust estimate of your model's performance by training and testing it on different subsets of the data.
- **Vietnamese:** Điều này thêm xác thực chéo vào đánh giá mô hình của bạn. Xác thực chéo cung cấp ước tính mạnh mẽ hơn về hiệu suất mô hình của bạn bằng cách huấn luyện và kiểm tra nó trên các tập hợp con khác nhau của dữ liệu.

Example 5: Saving Results to a File (Ví dụ 5: Lưu kết quả vào tệp)

```
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
results_file = os.path.join(data_path, 'predictions.csv')
results.to_csv(results_file, index=False)
print(f"Predictions saved to {results_file}")
```

- **English:** This shows how to save the actual and predicted values to a CSV file for further analysis or visualization.
- **Vietnamese:** Điều này cho thấy cách lưu các giá trị thực tế và dự đoán vào tệp CSV để phân tích hoặc trực quan hóa thêm.

Key Considerations:

- **Error Handling:** Always include error handling (e.g., `try...except` blocks) in your code to gracefully handle potential issues like missing data, invalid SMILES strings, or library import errors.
- **Data Cleaning:** Thoroughly clean and validate your data before any analysis. Check for missing values, outliers, and inconsistencies.
- **Feature Selection:** Experiment with different molecular descriptors and feature selection techniques to improve model performance.
- **Model Tuning:** Tune the hyperparameters of your chosen model using techniques like grid search or random search to find the optimal settings.
- **Documentation:** Document your code clearly and concisely, explaining the purpose of each step.

By following these guidelines and adapting the provided code snippets, you should be well-equipped to proceed with your Chembl 35 data analysis project. Remember to test your code thoroughly and adapt it to your specific research questions. Good luck!