PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Tìm target có hoạt tính đối kháng và đồng vận

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, chúng ta sẽ tập trung vào chủ đề "Topic_CheMBL_35_39" và giải quyết các vấn đề cụ thể bạn gặp phải.

1. Phân tích mô hình phân tích (Analysis Model):

Mục tiêu của chủ đề "Topic_CheMBL_35_39" là gì? Để đưa ra mô hình phân tích phù hợp, chúng ta cần xác định rõ câu hỏi nghiên cứu. Dưới đây là một số khả năng:

- **Dự đoán hoạt tính (Activity Prediction):** Xây dựng mô hình dự đoán hoạt tính của các hợp chất dựa trên cấu trúc hóa học của chúng (ví dụ: dự đoán IC50, Ki, EC50).
- **Phân tích SAR (Structure-Activity Relationship):** Tìm hiểu mối quan hệ giữa cấu trúc hóa học của các hợp chất và hoạt tính sinh học của chúng. Xác định các nhóm chức (functional groups) hoặc phần tử cấu trúc (structural motifs) quan trọng ảnh hưởng đến hoạt tính.
- **Phân cụm hợp chất (Compound Clustering):** Phân nhóm các hợp chất thành các cụm dựa trên tính tương đồng về cấu trúc hoặc hoạt tính.
- **Phân tích QSAR/QSPR (Quantitative Structure-Activity/Property Relationship):** Xây dựng mô hình định lượng mối quan hệ giữa cấu trúc hóa học và hoạt tính/tính chất của hợp chất.

Dựa trên mục tiêu cụ thể, chúng ta có thể lựa chọn các phương pháp phân tích phù hợp:

- **Machine Learning:** Các thuật toán như Random Forest, Support Vector Machines (SVM), Neural Networks, Gradient Boosting.
- **Cheminformatics:** Sử dụng các descriptor phân tử (ví dụ: Morgan fingerprints, physicochemical properties) để biểu diễn cấu trúc hóa học và phân tích.
- **Statistical Analysis:** Các phương pháp thống kê như hồi quy tuyến tính, phân tích phương sai (ANOVA).

2. Hướng dẫn song ngữ (Bilingual Guidance):

2.1. SQL:

- **Purpose:** To extract relevant data from the ChEMBL database.
- **Mục đích:** Trích xuất dữ liệu liên quan từ cơ sở dữ liệu ChEMBL.

2.2. Python:

- **Purpose:** To process the data, perform calculations, build models, and visualize results.
- **Mục đích:** Xử lý dữ liệu, thực hiện tính toán, xây dựng mô hình và trực quan hóa kết quả.

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

3.1. SQL (with error handling):

```sql
-- Extract data for Topic_CheMBL_35_39 (limited to 100 records)
-- Trích xuất dữ liệu cho Topic_CheMBL_35_39 (giới hạn 100 bản ghi)

SELECT
    act.molregno,
    act.standard_type,
```

```
    act.standard_value,
    act.standard_units,
    md.chembl_id,
    cs.canonical_smiles
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 IN ('IC50', 'Ki', 'EC50')  -- Example activity types
    AND act.standard_units = 'nM' -- Example units
    AND act.standard_value IS NOT NULL
    AND cs.canonical_smiles IS NOT NULL
    -- Handle the numeric check safely:
    AND act.standard_value::text ~ '^[0-9\.]+$'  -- Check if it's numeric
LIMIT 100;
```

Explanation:

- The `::text` cast is used to convert `standard_value` to text before applying the regular expression.
- We limit the results to 100 records using `LIMIT 100`.

3.2. Python (with scikit-learn version check):

```
import pandas as pd
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
import sklearn
import os

print(f"Scikit-learn version: {sklearn.__version__}")

# Define paths (following AIMLOps Template)
base_path = "../data"  # Adjust if needed
csv_file = os.path.join(base_path, "chembl_35_data.csv") # Replace with your actual
filename

# Load data
try:
    data = pd.read_csv(csv_file)
    print("Data loaded successfully.")
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}")
    exit()

# Data Cleaning and Preprocessing
data = data.dropna(subset=['canonical_smiles', 'standard_value']) # Drop rows with
missing SMILES or activity values
data = data[data['standard_value'].astype(str).str.match(r'^[0-9\.]+$')] #Keep numeric
values only
data['standard_value'] = pd.to_numeric(data['standard_value']) # Transform to numeric

# Function to convert SMILES to Morgan Fingerprints
```

```python
def smiles_to_morgan(smiles, radius=2, n_bits=2048):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=n_bits)
            return np.array(list(fp.ToBitString()), dtype=int)
        else:
            return None
    except:
        return None

# Apply the function to create fingerprints
data['morgan_fp'] = data['canonical_smiles'].apply(smiles_to_morgan)
data = data.dropna(subset=['morgan_fp'])  # Remove rows where fingerprint generation failed
data = data[data['morgan_fp'].map(len) > 0]

# Prepare data for machine learning
X = np.stack(data['morgan_fp'].values)
y = data['standard_value'].values

# 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)

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

# Make predictions
y_pred = model.predict(X_test)

# Calculate Mean Squared Error
if sklearn.__version__ < '0.22':
    mse = mean_squared_error(y_test, y_pred)
else:
    mse = mean_squared_error(y_test, y_pred, squared=False)  # Use squared=False for newer versions

print(f"Mean Squared Error: {mse}")
```

Explanation:

- **Error Handling:** Includes `try...except` blocks for file loading and SMILES conversion.
- **Scikit-learn Version Check:** Checks the scikit-learn version and uses the appropriate `mean_squared_error` call.
- **Data Preprocessing:** Handles missing data and converts SMILES to Morgan fingerprints.
- **Model Training:** Trains a Random Forest Regressor.
- **Evaluation:** Calculates and prints the Mean Squared Error.

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

Here are 5 examples of SQL and Python code snippets that you can adapt for your project.

Example 1: Extracting compounds with specific properties

- **SQL:** `sql SELECT md.chembl_id, 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`

```
md.mw_freebase < 300 -- Molecular weight less than 300     AND act.standard_type
= 'IC50'     LIMIT 10;
```

- **Python:** (Reading from a CSV file generated from the SQL query above) `python      import pandas as pd     data = pd.read_csv('../data/example1.csv') # Update with the correct path     print(data.head())`

Example 2: Calculating LogP using RDKit

- **SQL:** (Extract SMILES strings) `sql      SELECT md.chembl_id, cs.canonical_smiles FROM molecule_dictionary md     JOIN compound_structures cs ON md.molregno = cs.molregno     LIMIT 10;`

- **Python:**

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

data = pd.read_csv('../data/example2.csv') # Update with the correct path

def calculate_logp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        return Descriptors.MolLogP(mol)
    else:
        return None

data['logp'] = data['canonical_smiles'].apply(calculate_logp)
print(data.head())
```

Example 3: Filtering by activity value range

- **SQL:** `sql   SELECT md.chembl_id, 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_value BETWEEN 10 AND 100 -- IC50 between 10 and 100 nM   LIMIT 10;`

- **Python:** (Analyzing the extracted data)

```python
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('../data/example3.csv') # Update with the correct path
plt.hist(data['standard_value'])
plt.xlabel('IC50 (nM)')
plt.ylabel('Frequency')
plt.title('IC50 Distribution')
plt.show()
```

Example 4: Generating Morgan fingerprints and visualizing them

- **SQL:** (Extract SMILES) `sql     SELECT md.chembl_id, cs.canonical_smiles     FROM molecule_dictionary md     JOIN compound_structures cs ON md.molregno = cs.molregno     LIMIT 10;`

- **Python:**

```python
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
```

```
import matplotlib.pyplot as plt

data = pd.read_csv('../data/example4.csv') # Update with the correct path

def generate_morgan_fp(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprint(mol, 2) # Radius 2
        return fp
    else:
        return None

data['morgan_fp'] = data['canonical_smiles'].apply(generate_morgan_fp)
print(data.head())
```

Example 5: Basic Activity Prediction with a Simple Model

- **SQL:** `sql SELECT md.chembl_id, 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' LIMIT 100;`

- **Python:** (Simple Activity Prediction)

```
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

# Load data
data = pd.read_csv('../data/example5.csv') # Update with the correct path
data.dropna(inplace=True)

# RDKit: Generate Morgan Fingerprints
def generate_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is not None:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius=2, nBits=2048)
        return np.array(list(fp))
    else:
        return None

data['fingerprint'] = data['canonical_smiles'].apply(generate_fingerprint)
data.dropna(subset=['fingerprint'], inplace=True)

# Prepare data
X = np.stack(data['fingerprint'].values)
y = data['standard_value'].values

# Split data
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)
```

```
# Make predictions
y_pred = model.predict(X_test)

# Evaluate
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

Important Considerations:

- **Database Connection:** You'll need to establish a connection to your PostgreSQL database using a Python library like `psycopg2`.
- **Data Scaling/Normalization:** Consider scaling or normalizing your activity values (e.g., converting IC50 to pIC50) to improve model performance.
- **Feature Selection:** Explore different molecular descriptors and feature selection techniques to optimize your models.
- **Model Validation:** Use appropriate validation techniques (e.g., cross-validation) to ensure the robustness of your models.
- **Logging:** Implement logging to track the progress of your analysis and debug any issues.
- **AIMLOps:** Integrate your code into the AIMLOps framework for reproducibility, version control, and automated deployment.

Hãy cho tôi biết nếu bạn có bất kỳ câu hỏi cụ thể nào hoặc muốn tôi đi sâu hơn vào một khía cạnh cụ thể của dự án. Chúc bạn thành công!