PharmApp Suite

🧠 AI for Drug Discovery and Development 💉

🧩 Topic: Khai thác chất ức chế kinase từ ChEMBL

Tuyệt vời! Tôi rất vui được hỗ trợ bạn trong dự án phân tích dữ liệu ChEMBL 35 kết hợp RDKit. Với vai trò là một chuyên gia nghiên cứu và phát triển dược học, tôi sẽ cung cấp cho bạn những hướng dẫn chi tiết, code mẫu SQL và Python, cũng như các ví dụ cụ thể để bạn có thể triển khai dự án một cách hiệu quả.

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

Dựa trên thông tin bạn cung cấp, tôi hiểu rằng bạn đang muốn:

- **Trích xuất dữ liệu từ cơ sở dữ liệu ChEMBL 35:** Sử dụng SQL để truy vấn và lọc dữ liệu liên quan đến hoạt tính sinh học của các hợp chất.
- **Tiền xử lý dữ liệu:** Sử dụng Python và RDKit để xử lý dữ liệu, bao gồm chuyển đổi SMILES sang fingerprint, tính toán các thuộc tính vật lý hóa học, và xử lý các giá trị bị thiếu.
- **Phân tích dữ liệu:** Sử dụng các mô hình thống kê và học máy để khám phá các mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học.
- **Lưu trữ và quản lý dữ liệu:** Sử dụng cấu trúc thư mục AIMLOps để quản lý code, dữ liệu và kết quả.

Mô hình phân tích đề xuất:

Tôi đề xuất một quy trình phân tích gồm các bước sau:

1. **Data Extraction and Cleaning:**
    - **SQL:** Extract relevant data from ChEMBL 35 (e.g., compounds, activities, targets).
    - **Python:** Handle missing values, convert data types, and filter out irrelevant entries.
2. **Feature Engineering:**
    - **RDKit:** Generate molecular fingerprints (e.g., Morgan fingerprints) and calculate physicochemical properties (e.g., LogP, MW, TPSA).
3. **Data Analysis:**
    - **Exploratory Data Analysis (EDA):** Visualize data distributions, identify outliers, and explore relationships between variables.
    - **Model Building:** Train machine learning models to predict activity based on molecular features. Potential models include:
        - **Regression Models:** Linear Regression, Random Forest Regression, Support Vector Regression.
        - **Classification Models:** Logistic Regression, Random Forest Classification, Support Vector Classification.
4. **Model Evaluation:**
    - Evaluate model performance using appropriate metrics (e.g., R-squared, RMSE for regression; accuracy, precision, recall, F1-score for classification).
    - Use cross-validation to ensure model generalizability.
5. **Interpretation and Reporting:**
    - Interpret model results to identify important features and understand the structure-activity relationship.
    - Generate reports and visualizations to communicate findings.

2. Hướng dẫn song ngữ (Bilingual Instructions)

English:

This project aims to analyze ChEMBL 35 data to understand the relationship between chemical structure and biological activity. We will use SQL to extract data, RDKit to generate molecular features, and machine learning models to predict activity. The AIMLOps structure will be used to organize code, data, and results.

Tiếng Việt:

Dự án này nhằm mục đích phân tích dữ liệu ChEMBL 35 để hiểu mối quan hệ giữa cấu trúc hóa học và hoạt tính sinh học. Chúng ta sẽ sử dụng SQL để trích xuất dữ liệu, RDKit để tạo ra các đặc trưng phân tử và các mô hình học máy để dự đoán hoạt tính. Cấu trúc AIMLOps sẽ được sử dụng để tổ chức code, dữ liệu và kết quả.

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

SQL (English):

```sql
-- Extract 100 compounds with activity data for a specific target
SELECT DISTINCT
    md.molregno,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Acetylcholinesterase'  -- Replace with your target of interest
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND act.standard_value::text ~ '^[0-9\.]+$' -- Ensure standard_value is numeric
LIMIT 100;
```

SQL (Tiếng Việt):

```sql
-- Trích xuất 100 hợp chất có dữ liệu hoạt tính cho một mục tiêu cụ thể
SELECT DISTINCT
    md.molregno,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.pref_name = 'Acetylcholinesterase'  -- 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 act.standard_value::text ~ '^[0-9\.]+$' -- Đảm bảo standard_value là số
LIMIT 100;
```

Lưu ý: Đoạn code SQL trên đã được sửa để khắc phục lỗi `ERROR: operator does not exist: numeric ~ unknown`. Chúng ta đã chuyển đổi `act.standard_value` sang kiểu text và sử dụng toán tử `~` để so sánh với một biểu thức chính quy.

Python (English):

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

# Define the base path for your project
base_path = "path/to/your/project" # Replace with your actual path

# Load the data from the CSV file
data_path = os.path.join(base_path, "data", "chembl_data.csv") # Replace
"chembl_data.csv" with your actual filename
df = pd.read_csv(data_path)

# Handle missing values (replace with the mean)
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df['standard_value'] = df['standard_value'].fillna(df['standard_value'].mean())

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

# Function to calculate Morgan fingerprints using RDKit
def calculate_morgan_fingerprint(smiles, radius=2, nBits=2048):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits)
        return np.array(list(fp.ToBitString()), dtype=int)
    else:
        return None

# Apply the function to create a new column with fingerprints
df['fingerprint'] = df['canonical_smiles'].apply(calculate_morgan_fingerprint)

# Drop rows where fingerprint calculation failed
df = df.dropna(subset=['fingerprint'])

# Convert fingerprints to a matrix
X = np.vstack(df['fingerprint'].to_numpy())
y = df['pIC50'].to_numpy()

# Data scaling
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

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

# Make predictions on the test set
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}")
```

Python (Tiếng Việt):

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

# Xác định đường dẫn gốc của dự án
base_path = "path/to/your/project" # Thay thế bằng đường dẫn thực tế của bạn

# Tải dữ liệu từ file CSV
data_path = os.path.join(base_path, "data", "chembl_data.csv") # Thay thế
"chembl_data.csv" bằng tên file thực tế của bạn
df = pd.read_csv(data_path)

# Xử lý giá trị bị thiếu (thay thế bằng giá trị trung bình)
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')
df['standard_value'] = df['standard_value'].fillna(df['standard_value'].mean())

# Chuyển đổi IC50 sang pIC50
df['pIC50'] = -np.log10(df['standard_value'] / 1e9)

# Hàm tính toán Morgan fingerprint sử dụng RDKit
def calculate_morgan_fingerprint(smiles, radius=2, nBits=2048):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits)
        return np.array(list(fp.ToBitString()), dtype=int)
    else:
        return None

# Áp dụng hàm để tạo một cột mới với fingerprint
df['fingerprint'] = df['canonical_smiles'].apply(calculate_morgan_fingerprint)

# Loại bỏ các hàng mà tính toán fingerprint thất bại
df = df.dropna(subset=['fingerprint'])
```

```python
# Chuyển đổi fingerprint thành ma trận
X = np.vstack(df['fingerprint'].to_numpy())
y = df['pIC50'].to_numpy()

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
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 Regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
y_pred = model.predict(X_test)

# Đánh giá mô hình
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: Đoạn code Python trên đã được cập nhật để:

- Sử dụng `StandardScaler` để chuẩn hóa dữ liệu trước khi huấn luyện mô hình.
- Sử dụng `mean_squared_error` và `r2_score` từ `sklearn.metrics` để đánh giá mô hình.
- Loại bỏ tham số `squared=False` trong hàm `mean_squared_error` (nếu bạn đang sử dụng phiên bản scikit-learn cũ, hãy nâng cấp lên phiên bản mới nhất).
- Xử lý giá trị bị thiếu bằng cách thay thế chúng bằng giá trị trung bình của cột `standard_value`.
- Chuyển đổi giá trị IC50 sang pIC50 bằng công thức `pIC50 = -np.log10(IC50 / 1e9)`.

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

Ví dụ 1: Trích xuất dữ liệu cho một họ protein cụ thể (Extract data for a specific protein family)

SQL (English):

```sql
-- Extract compounds active against a specific protein family (e.g., Kinases)
SELECT DISTINCT
    md.molregno,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
JOIN
    component_sequences cs2 ON td.component_id = cs2.component_id
WHERE
    cs2.protein_family_desc LIKE '%Kinase%'
```

```
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND act.standard_value::text ~ '^[0-9\.]+$'
LIMIT 100;
```

SQL (Tiếng Việt):

```
-- Trích xuất các hợp chất có hoạt tính chống lại một họ protein cụ thể (ví dụ:
Kinase)
SELECT DISTINCT
    md.molregno,
    cs.canonical_smiles,
    act.standard_value,
    act.standard_units,
    act.standard_type
FROM
    molecule_dictionary md
JOIN
    compound_structures cs ON md.molregno = cs.molregno
JOIN
    activities act ON md.molregno = act.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
JOIN
    component_sequences cs2 ON td.component_id = cs2.component_id
WHERE
    cs2.protein_family_desc LIKE '%Kinase%'
    AND act.standard_type = 'IC50'
    AND act.standard_units = 'nM'
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0
    AND act.standard_value::text ~ '^[0-9\.]+$'
LIMIT 100;
```

Ví dụ 2: Tính toán các thuộc tính vật lý hóa học (Calculate physicochemical properties)

Python (English):

```
from rdkit import Chem
from rdkit.Chem import Descriptors

def calculate_physicochemical_properties(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        mw = Descriptors.MolWt(mol)
        logp = Descriptors.MolLogP(mol)
        hbd = Descriptors.NumHDonors(mol)
        hba = Descriptors.NumHAcceptors(mol)
        tpsa = Descriptors.TPSA(mol)
        return mw, logp, hbd, hba, tpsa
    else:
        return None, None, None, None, None

df[['MW', 'LogP', 'HBD', 'HBA', 'TPSA']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_physicochemical_properties(x)))
```

Python (Tiếng Việt):

```
from rdkit import Chem
from rdkit.Chem import Descriptors
```

```
def calculate_physicochemical_properties(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        mw = Descriptors.MolWt(mol)
        logp = Descriptors.MolLogP(mol)
        hbd = Descriptors.NumHDonors(mol)
        hba = Descriptors.NumHAcceptors(mol)
        tpsa = Descriptors.TPSA(mol)
        return mw, logp, hbd, hba, tpsa
    else:
        return None, None, None, None, None

df[['MW', 'LogP', 'HBD', 'HBA', 'TPSA']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_physicochemical_properties(x)))
```

Ví dụ 3: Xây dựng mô hình hồi quy tuyến tính (Build a linear regression model)

Python (English):

```
from sklearn.linear_model import LinearRegression

# Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
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}")
```

Python (Tiếng Việt):

```
from sklearn.linear_model import LinearRegression

# Huấn luyện mô hình Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
y_pred = model.predict(X_test)

# Đánh giá mô hình
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}")
```

Ví dụ 4: Sử dụng PCA để giảm chiều dữ liệu (Use PCA for dimensionality reduction)

Python (English):

```
from sklearn.decomposition import PCA

# Apply PCA to reduce the number of features
```

```
pca = PCA(n_components=100)  # Reduce to 100 components
X_pca = pca.fit_transform(X)

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

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

# Make predictions on the test set
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}")
```

Python (Tiếng Việt):

```
from sklearn.decomposition import PCA

# Áp dụng PCA để giảm số lượng đặc trưng
pca = PCA(n_components=100)  # Giảm xuống còn 100 thành phần
X_pca = pca.fit_transform(X)

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2,
random_state=42)

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

# Dự đoán trên tập kiểm tra
y_pred = model.predict(X_test)

# Đánh giá mô hình
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}")
```

Ví dụ 5: Sử dụng Grid Search để tối ưu hóa siêu tham số (Use Grid Search for hyperparameter optimization)

Python (English):

```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10]
}
```

```
# Create a GridSearchCV object
grid_search = GridSearchCV(estimator=RandomForestRegressor(random_state=42),
                           param_grid=param_grid,
                           cv=3,
                           scoring='neg_mean_squared_error',
                           verbose=2)

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Print the best parameters and the best score
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best score: {-grid_search.best_score_}")

# Use the best model to make predictions
best_model = grid_search.best_estimator_
y_pred = best_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}")
```

Python (Tiếng Việt):

```
from sklearn.model_selection import GridSearchCV

# Xác định lưới tham số
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10]
}

# Tạo đối tượng GridSearchCV
grid_search = GridSearchCV(estimator=RandomForestRegressor(random_state=42),
                           param_grid=param_grid,
                           cv=3,
                           scoring='neg_mean_squared_error',
                           verbose=2)

# Fit grid search vào dữ liệu
grid_search.fit(X_train, y_train)

# In ra các tham số tốt nhất và điểm số tốt nhất
print(f"Tham số tốt nhất: {grid_search.best_params_}")
print(f"Điểm số tốt nhất: {-grid_search.best_score_}")

# Sử dụng mô hình tốt nhất để đưa ra dự đoán
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Đánh giá mô hình
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}")
```

Hy vọng những hướng dẫn và ví dụ này sẽ giúp bạn triển khai dự án phân tích dữ liệu ChEMBL 35 một cách thành công. Nếu bạn có bất kỳ câu hỏi nào khác, đừng ngần ngại hỏi!