PharmApp Suite

🧠 AI for Drug Discovery and Development 🧪

🧩 Topic: Predicting drug-target interactions using ML

Okay, I understand. You're working on a Chembl 35 data analysis project within an AIMLops framework, focusing on a specific topic (Topic_CheMBL_35_70). You have your data in a PostgreSQL database and plan to use RDKit and Jupyter Notebooks for analysis. You need assistance with the code, specifically addressing the SQL error and the scikit-learn version issue. You want to retrieve a limited sample (100 rows) for testing and provide 5 illustrative examples of your analysis.

Here's a breakdown of my proposed solution, incorporating both English and Chinese (simplified) explanations, code examples, and error resolution:

1. Analysis of the Model and Approach

The project leverages a standard drug discovery data workflow, querying the ChEMBL database, and then processing and analyzing the results using RDKit and scikit-learn. This approach often includes the following steps:

- **Data Retrieval (SQL):** Querying the ChEMBL database to retrieve relevant bioactivity data (e.g., IC50, Ki values) along with compound information (SMILES strings).
- **Data Preprocessing (Python/RDKit):** Cleaning the data (removing invalid entries, handling missing values), converting SMILES strings to molecular objects, and generating molecular descriptors.
- **Data Analysis (Python/scikit-learn):** Building models to predict bioactivity based on molecular descriptors. This might involve regression models (e.g., linear regression, random forest) or classification models (e.g., logistic regression, support vector machines).

Analysis of the errors:

- **SQL Error:** `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'` This indicates an issue with the regular expression matching on the `standard_value` column. PostgreSQL likely doesn't support regular expression matching directly on numeric columns, and you're trying to use the ~ operator which is for string matching.
- **Scikit-learn Error:** `old scikit-learn version does not support parameters squared=False in the mean_squared_error function` This implies that you are using an older version of scikit-learn. The `squared=False` parameter for `mean_squared_error` was introduced in a later version.

Chinese Explanation (简化字):

这个项目利用标准的药物发现数据流程，查询 ChEMBL 数据库，然后使用 RDKit 和 scikit-learn 处理和分析结果。 这种方法通常包括以下步骤：

- 数据检索 **(SQL):** 查询 ChEMBL 数据库以检索相关的生物活性数据（例如，IC50，Ki 值）以及化合物信息（SMILES 字符串）。
- 数据预处理 **(Python/RDKit):** 清理数据（删除无效条目，处理缺失值），将 SMILES 字符串转换为分子对象，并生成分子描述符。
- 数据分析 **(Python/scikit-learn):** 构建模型以基于分子描述符预测生物活性。 这可能涉及回归模型（例如，线性回归，随机森林）或分类模型（例如，逻辑回归，支持向量机）。

错误分析:

- **SQL 错误:** `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'` 这表明 `standard_value` 列上的正则表达式匹配存在问题。 PostgreSQL 可能不支持直接在数值列上进行正则表达式匹配，并且您正在尝试使用~运算符，该运算符用于字符串匹配。
- **Scikit-learn 错误:** `old scikit-learn version does not support parameters squared=False in the mean_squared_error function` 这意味着您正在使用旧版本的 scikit-learn。 `mean_squared_error` 的 `squared=False` 参数是在更高版本中引入的。

2. SQL Code (to retrieve data and save to CSV)

```sql
-- Topic_CheMBL_35_70.sql
-- Retrieve 100 rows of bioactivity data related to a specific target/protein.
-- Replace "target_chembl_id" with the actual ChEMBL ID of your target of interest.

SELECT
    act.activity_id,
    cmp.chembl_id,
    cmp.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    act.pchembl_value
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
JOIN
    target_dictionary tgt ON act.tid = tgt.tid
WHERE
    tgt.chembl_id = 'CHEMBL205'  -- Replace with your target ChEMBL ID
(Example:CHEMBL205 for Carbonic Anhydrase II)
    AND act.standard_type = 'IC50'  -- Filter for IC50 values (Example)
    AND act.standard_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$' -- Convert to text before regex
matching
    AND act.standard_units = 'nM' --Filter for nM values (Example)
    AND act.pchembl_value IS NOT NULL --Filter for pchembl value is not null (Example)
LIMIT 100;

--Save the results to a CSV file:
--\copy (SELECT * FROM your_query) TO 'path/to/your/data/Topic_CheMBL_35_70.csv' WITH
CSV HEADER; -- Run this in psql command line after connecting to database
```

Explanation (SQL):

- The SQL query joins three tables: `activities`, `molecule_dictionary`, and `target_dictionary` to retrieve the necessary information (activity, molecule details, and target details).
- The `WHERE` clause filters the data based on your target ChEMBL ID, activity type (e.g., IC50), and other criteria.
- **Important:** The line `AND act.standard_value::text ~ '^[0-9\.]+$'` fixes the error. It converts the `standard_value` column to `text` type before performing the regular expression match. This is crucial because PostgreSQL doesn't directly support regular expression matching on numeric columns.
- `LIMIT 100` limits the result set to 100 rows.
- The last line provide the way to export to csv, but it only run on psql command line

Chinese Explanation (SQL):

```sql
-- Topic_CheMBL_35_70.sql
-- 检索与特定靶标/蛋白质相关的100行生物活性数据。
-- 将“target_chembl_id”替换为您感兴趣的靶标的实际ChEMBL ID。

SELECT
    act.activity_id,
    cmp.chembl_id,
    cmp.canonical_smiles,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    act.pchembl_value
FROM
    activities act
JOIN
    molecule_dictionary cmp ON act.molregno = cmp.molregno
JOIN
    target_dictionary tgt ON act.tid = tgt.tid
WHERE
    tgt.chembl_id = 'CHEMBL205'  -- 替换为您的靶标ChEMBL ID（示例：CHEMBL205为碳酸酐酶
II）
    AND act.standard_type = 'IC50'  -- 过滤IC50值（示例）
    AND act.standard_value IS NOT NULL
    AND act.standard_value::text ~ '^[0-9\.]+$' -- 在正则表达式匹配之前转换为文本
    AND act.standard_units = 'nM' -- 过滤nM值（示例）
    AND act.pchembl_value IS NOT NULL -- 过滤pchembl值不为空的(示例)
LIMIT 100;

-- 将结果保存到CSV文件:
--\copy (SELECT * FROM your_query) TO 'path/to/your/data/Topic_CheMBL_35_70.csv' WITH
CSV HEADER; -- 在连接到数据库后，在psql命令行中运行此命令
```

3. Python Code (Jupyter Notebook - Topic_CheMBL_35_70_1_Data_Preprocessing.ipynb)

```python
# Topic_CheMBL_35_70_1_Data_Preprocessing.ipynb
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np

# Define base path
base_path = "../data"  # Adjust this to your project's base path

# Construct the file path
csv_file_path = os.path.join(base_path, "Topic_CheMBL_35_70.csv")

# Load the data from CSV
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file_path}.  Make sure the SQL query has
been run and the CSV file created.")
    exit()

print(f"Loaded {len(df)} rows from {csv_file_path}")

# Data Cleaning and Preprocessing
def clean_data(df):
    """
```

```
    Cleans the DataFrame by:
        - Removing rows with missing SMILES or standard_value.
        - Converting standard_value to numeric.
    """
    df = df.dropna(subset=['canonical_smiles', 'standard_value'])
    df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce') #
Convert and handle errors
    df = df.dropna(subset=['standard_value']) # Drop rows where conversion failed
    df = df[df['standard_units'] == 'nM']  # Keep only nM values
    df = df.drop_duplicates(subset=['canonical_smiles'])  #Drop duplicates
    return df

df = clean_data(df)
print(f"DataFrame size after cleaning: {len(df)} rows")

# RDKit Mol Object Creation
def create_mol_objects(df):
    """
    Creates RDKit Mol objects from SMILES strings.
    """
    df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))
    df = df[df['mol'].notna()] #Remove entry where SMILES not valid
    return df

df = create_mol_objects(df)
print(f"DataFrame size after Mol object creation: {len(df)} rows")

# Example: Display the first 5 rows
print(df.head())

# Save the processed dataframe (optional)
processed_file_path = os.path.join(base_path, "Topic_CheMBL_35_70_processed.csv")
df.to_csv(processed_file_path, index=False)

print(f"Processed data saved to {processed_file_path}")
```

Explanation (Python - Data Preprocessing):

- **Imports:** Imports necessary libraries like `os`, `pandas`, `rdkit.Chem`, and `rdkit.Chem.Descriptors`.
- **File Handling:** Uses `os.path.join` to create the file path for the CSV data file, ensuring cross-platform compatibility. The code also includes error handling for the file not being found.
- **Data Loading:** Loads the CSV data into a pandas DataFrame.
- **Data Cleaning:** Removes rows with missing SMILES strings or activity values. Converts the 'standard_value' column to numeric, handling potential errors. Filters to retain only data with 'nM' units.
- **RDKit Mol Object Creation:** Creates RDKit molecule objects from the SMILES strings using `Chem.MolFromSmiles()`. Invalid SMILES strings will result in `None` values in the 'mol' column, and these rows are removed.
- **Output:** Prints the head of the resulting DataFrame and saves processed data to another file.

Chinese Explanation (Python - 数据预处理):

```
# Topic_CheMBL_35_70_1_Data_Preprocessing.ipynb
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
import numpy as np
```

```
# 定义基本路径
base_path = "../data"  # 将此调整为您的项目基本路径

# 构建文件路径
csv_file_path = os.path.join(base_path, "Topic_CheMBL_35_70.csv")

# 从CSV加载数据
try:
    df = pd.read_csv(csv_file_path)
except FileNotFoundError:
    print(f"错误：在{csv_file_path}找不到文件。请确保已运行SQL查询并创建了CSV文件。")
    exit()

print(f"从{csv_file_path}加载了{len(df)}行")

# 数据清洗和预处理
def clean_data(df):
    """
    通过以下方式清理DataFrame：
        - 删除缺少SMILES或standard_value的行。
        - 将standard_value转换为数字。
    """
    df = df.dropna(subset=['canonical_smiles', 'standard_value'])
    df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce') # 转换
并处理错误
    df = df.dropna(subset=['standard_value']) # 删除转换失败的行
    df = df[df['standard_units'] == 'nM']  # 仅保留nM值
    df = df.drop_duplicates(subset=['canonical_smiles'])  # 删除重复项
    return df

df = clean_data(df)
print(f"清洗后DataFrame的大小：{len(df)}行")

# RDKit Mol对象创建
def create_mol_objects(df):
    """
    从SMILES字符串创建RDKit Mol对象。
    """
    df['mol'] = df['canonical_smiles'].apply(lambda x: Chem.MolFromSmiles(x))
    df = df[df['mol'].notna()] # 删除SMILES无效的条目
    return df

df = create_mol_objects(df)
print(f"创建Mol对象后DataFrame的大小：{len(df)}行")

# 示例：显示前5行
print(df.head())

# 保存已处理的数据框（可选）
processed_file_path = os.path.join(base_path, "Topic_CheMBL_35_70_processed.csv")
df.to_csv(processed_file_path, index=False)

print(f"已处理的数据保存到{processed_file_path}")
```

4. Python Code (Jupyter Notebook - Topic_CheMBL_35_70_2_Descriptor_Calculation_and_Modeling.ipynb)

```python
# Topic_CheMBL_35_70_2_Descriptor_Calculation_and_Modeling.ipynb

import os
import pandas as pd
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
from sklearn.preprocessing import StandardScaler
import numpy as np

# Define base path
base_path = "../data"

# Construct file path for the processed data
processed_file_path = os.path.join(base_path, "Topic_CheMBL_35_70_processed.csv")

# Load the processed data
try:
    df = pd.read_csv(processed_file_path)
except FileNotFoundError:
    print(f"Error: File not found at {processed_file_path}.  Make sure the data
preprocessing notebook has been run.")
    exit()

# Descriptor Calculation
def calculate_descriptors(mol):
    """Calculates a set of RDKit descriptors for a molecule."""
    descriptors = {}
    descriptors['MW'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
    descriptors['HBD'] = Descriptors.NumHDonors(mol)
    return descriptors

df['descriptors'] = df['mol'].apply(calculate_descriptors)

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

# Data preparation for modeling
X = df[['MW', 'LogP', 'HBA', 'HBD']].fillna(0) # Handle any potential NaN values
y = df['pchembl_value']

# Data Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, 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)
try:
    mse = mean_squared_error(y_test, y_pred)
except TypeError:
     y_test = y_test.astype(float)
     y_pred = y_pred.astype(float)
     mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

# Display results
print("Model Coefficients:", model.coef_)
print("Model Intercept:", model.intercept_)
```

Explanation (Python - Descriptor Calculation and Modeling):

- **Imports:** Imports necessary libraries from `rdkit` and `scikit-learn`.
- **File Loading:** Loads the processed data from the CSV file created in the previous notebook.
- **Descriptor Calculation:** Defines a function to calculate a set of molecular descriptors (Molecular Weight, LogP, Hydrogen Bond Acceptors, Hydrogen Bond Donors) using RDKit.
- **Data Preparation:** Extracts the calculated descriptors and the target variable ('pchembl_value') into X and y. Fills any potential NaN values in the descriptor columns with 0.
- **Data Scaling:** Scales the features using `StandardScaler` to have zero mean and unit variance. This is often important for linear models.
- **Train/Test Split:** Splits the data into training and testing sets using `train_test_split`.
- **Model Training:** Trains a linear regression model using the training data.
- **Model Evaluation:** Predicts activity values for the test set and calculates the mean squared error.
- **Scikit-learn Error Handling:** The original error involved `squared=False` parameter, it has been removed.
- **Output:** Prints the Mean Squared Error, Model Coefficients, and Intercept.

Chinese Explanation (Python - 描述符计算和建模):

```
# Topic_CheMBL_35_70_2_Descriptor_Calculation_and_Modeling.ipynb

import os
import pandas as pd
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
from sklearn.preprocessing import StandardScaler
import numpy as np

# 定义基本路径
base_path = "../data"

# 构建已处理数据的路径
processed_file_path = os.path.join(base_path, "Topic_CheMBL_35_70_processed.csv")

# 加载已处理的数据
try:
    df = pd.read_csv(processed_file_path)
except FileNotFoundError:
    print(f"错误：在{processed_file_path}找不到文件。请确保已运行数据预处理笔记本。")
    exit()
```

```python
# 描述符计算
def calculate_descriptors(mol):
    """计算分子的RDKit描述符集合。"""
    descriptors = {}
    descriptors['MW'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
    descriptors['HBD'] = Descriptors.NumHDonors(mol)
    return descriptors

df['descriptors'] = df['mol'].apply(calculate_descriptors)

# 将描述符转换为列
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

# 准备建模数据
X = df[['MW', 'LogP', 'HBA', 'HBD']].fillna(0)  # 处理任何潜在的NaN值
y = df['pchembl_value']

# 数据缩放
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 将数据拆分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# 模型训练
model = LinearRegression()
model.fit(X_train, y_train)

# 模型评估
y_pred = model.predict(X_test)
try:
    mse = mean_squared_error(y_test, y_pred)
except TypeError:
     y_test = y_test.astype(float)
     y_pred = y_pred.astype(float)
     mse = mean_squared_error(y_test, y_pred)
print(f"均方误差: {mse}")

# 显示结果
print("模型系数: ", model.coef_)
print("模型截距: ", model.intercept_)
```

5. 5 Examples (using the calculated data)

Here are 5 examples of analyses you could perform, building on the code above:

Example 1: Scatter Plot of LogP vs. pChEMBL Value

```python
import matplotlib.pyplot as plt
plt.scatter(df['LogP'], df['pchembl_value'])
plt.xlabel('LogP')
plt.ylabel('pChEMBL Value')
plt.title('LogP vs. pChEMBL Value')
plt.show()
```

Example 2: Distribution of Molecular Weights

```
import seaborn as sns
sns.histplot(df['MW'])
plt.xlabel('Molecular Weight')
plt.title('Distribution of Molecular Weights')
plt.show()
```

Example 3: Calculate and Visualize TPSA (Topological Polar Surface Area)

```
from rdkit.Chem import rdMolDescriptors

def calculate_tpsa(mol):
    return rdMolDescriptors.CalcTPSA(mol)

df['TPSA'] = df['mol'].apply(calculate_tpsa)

plt.scatter(df['TPSA'], df['pchembl_value'])
plt.xlabel('TPSA')
plt.ylabel('pChEMBL Value')
plt.title('TPSA vs. pChEMBL Value')
plt.show()
```

Example 4: Build a Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor

# Model Training
model = RandomForestRegressor(n_estimators=100, random_state=42) # You can adjust
hyperparameters
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Random Forest Mean Squared Error: {mse}")
```

Example 5: Examining the Most Important Features in the Random Forest Model

```
# Get feature importances from the trained Random Forest model
importances = model.feature_importances_

# Create a dictionary to map feature names to importances
feature_importances = dict(zip(['MW', 'LogP', 'HBA', 'HBD'], importances))

# Print the feature importances
print("Feature Importances:")
for feature, importance in feature_importances.items():
    print(f"{feature}: {importance}")
```

Chinese Explanation (5 Examples):

这里有 5 个可以执行的分析示例，基于上面的代码：

示例 **1: LogP** 与 **pChEMBL** 值的散点图

```
import matplotlib.pyplot as plt
plt.scatter(df['LogP'], df['pchembl_value'])
plt.xlabel('LogP')
plt.ylabel('pChEMBL 值')
plt.title('LogP 与 pChEMBL 值')
plt.show()
```

示例 **2:** 分子量分布

```
import seaborn as sns
sns.histplot(df['MW'])
plt.xlabel('分子量')
plt.title('分子量分布')
plt.show()
```

示例 **3:** 计算并可视化 **TPSA (**拓扑极性表面积**)**

```
from rdkit.Chem import rdMolDescriptors

def calculate_tpsa(mol):
    return rdMolDescriptors.CalcTPSA(mol)

df['TPSA'] = df['mol'].apply(calculate_tpsa)

plt.scatter(df['TPSA'], df['pchembl_value'])
plt.xlabel('TPSA')
plt.ylabel('pChEMBL 值')
plt.title('TPSA 与 pChEMBL 值')
plt.show()
```

示例 **4:** 构建随机森林回归模型

```
from sklearn.ensemble import RandomForestRegressor

# 模型训练
model = RandomForestRegressor(n_estimators=100, random_state=42) # 您可以调整超参数
model.fit(X_train, y_train)

# 模型评估
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"随机森林均方误差: {mse}")
```

示例 **5:** 检查随机森林模型中最重要的特征

```
# 从训练好的随机森林模型中获取特征重要性
importances = model.feature_importances_

# 创建一个字典，将特征名称映射到重要性
feature_importances = dict(zip(['MW', 'LogP', 'HBA', 'HBD'], importances))

# 打印特征重要性
print("特征重要性:")
for feature, importance in feature_importances.items():
    print(f"{feature}: {importance}")
```

Key Improvements and Considerations:

- **Error Handling:** Addresses the SQL error by explicitly casting the `standard_value` to text before performing the regular expression match. Removes the need for `squared=False` parameter for `mean_squared_error`
- **File Path Management:** Uses `os.path.join` for robust file path construction.
- **Data Cleaning:** Includes data cleaning steps to handle missing values and invalid SMILES strings.
- **Descriptor Calculation:** Provides a clear example of how to calculate molecular descriptors using RDKit.
- **Modeling:** Demonstrates a simple linear regression model and Random Forest Model.

- **Comments and Explanations:** Includes detailed comments and explanations in both English and Chinese.
- **Modular Code:** The code is divided into functions for better readability and reusability.
- **Example Analyses:** Provides 5 examples to get you started with exploring your data.
- **AIMLops Structure:** The notebook names (`Topic_CheMBL_35_70_1_...`, `Topic_CheMBL_35_70_2_...`) follow the specified convention. The file structure assumes a `../data` directory relative to the notebooks.

Remember to install the necessary libraries: `pip install pandas rdkit scikit-learn matplotlib seaborn`.

This comprehensive guide should help you get started with your Chembl 35 data analysis project. Let me know if you have any more questions or need further assistance!