PharmApp Suite

🧠 AI for Drug Discovery and Development 🧪

🧩 Topic: Dimensionality reduction for chemical descriptors

Okay, I understand. You're working on a Chembl 35 data analysis project using RDKit and you're following an AIMLops-inspired folder structure. You're running into some issues with your SQL query and an outdated scikit-learn version. Let's break this down and build a solution.

1. Analysis of the Analysis Model

Your project aims to extract and analyze data from the Chembl 35 database to support drug research and development. Here's a potential analysis model, keeping in mind your limited sample size (100 rows):

- **Data Extraction:** Use SQL queries to retrieve relevant data from the Chembl 35 database (e.g., bioactivity data, molecule structures). You want to focus on specific targets, assays, or activity types.
- **Data Preprocessing:** Clean and prepare the data using Python and RDKit. This includes:
    - Handling missing values.
    - Converting SMILES strings to RDKit molecule objects.
    - Calculating molecular descriptors (e.g., molecular weight, LogP, TPSA).
    - Filtering based on activity values (e.g., selecting compounds with IC50 values below a certain threshold).
- **Exploratory Data Analysis (EDA):** Visualize and explore the data to identify trends and relationships.
    - Histograms of activity values.
    - Scatter plots of molecular descriptors vs. activity.
    - Box plots to compare activities across different compound classes.
- **Simple Modeling (Optional):** Due to the small sample size, complex modeling might not be ideal. However, you could try a simple model to establish a baseline and understand limitations.
    - Linear Regression: Predicting activity based on molecular descriptors.
    - Consider using cross-validation to get a better estimate of model performance.
- **Interpretation and Reporting:** Summarize the findings and draw conclusions about the relationships between molecular properties and activity. Report any challenges encountered during data processing or modeling.

Key Considerations:

- **Limited Data:** With only 100 rows, you are severely limited in the types of analyses you can perform and the conclusions you can draw. Focus on demonstrating your workflow and identifying potential areas for further investigation with a larger dataset.
- **Target Selection:** Choose a specific target or assay to focus your analysis on. This will make your results more meaningful.
- **Data Quality:** Be aware that the Chembl 35 database contains data from various sources, and data quality can vary.

2. SQL and Python Code

Here's the SQL and Python code with the corrections and adhering to your file structure:

SQL (`../data/chembl35_data.sql`)

```sql
-- Select bioactivity data for a specific target (replace 'CHEMBL205' with your target
of interest - EGFR Example)
-- Limiting to 100 rows for demonstration purposes.

SELECT
    act.molregno,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    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.confidence_score >= 9  -- Ensure high confidence data
    AND act.standard_type = 'IC50' -- Focus on IC50 values
    AND act.standard_units = 'nM'  -- Ensure consistent units
    AND act.standard_value IS NOT NULL
    AND act.standard_value::TEXT ~ '^[0-9\.]+$' -- Corrected regular expression
handling (explicit type casting)
    AND act.target_chembl_id = 'CHEMBL205'  -- Focus on a specific target (EGFR -
Example)
LIMIT 100;

-- Save the result to a CSV file using pgAdmin or \copy command.  Example:
-- \copy (SELECT ...) TO 'path/to/your/data/chembl35_data.csv' WITH CSV HEADER;
```

Explanation of SQL Changes:

- **Error Correction:** The error `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'` arises because PostgreSQL's ~ operator (regular expression match) might not directly work on numeric columns. The fix is to explicitly cast the `standard_value` to `TEXT` using `act.standard_value::TEXT`.
- **Target Filtering:** Added `AND act.target_chembl_id = 'CHEMBL205'` to filter for a specific target (EGFR in this example). *Crucially, you MUST replace `'CHEMBL205'` with the Chembl ID of the target you want to analyze.*
- **Confidence Score:** Included `act.confidence_score >= 9` to ensure you're working with high-quality data.
- **LIMIT Clause:** Added `LIMIT 100` to restrict the number of rows.
- **Saving to CSV:** The comment at the end shows how to save the SQL query result directly to a CSV file using `\copy`. You will execute this in pgAdmin after running the select statement.

Python Code (`notebook/Topic_CheMBL_35_69_1_data_processing.ipynb`)

```python
import pandas as pd
import os
from rdkit import Chem
from rdkit.Chem import Descriptors
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 the base path
base_path = os.path.abspath(os.path.join(os.getcwd(), "..")) # Assuming notebook is in
the 'notebook' directory

# Data Loading
data_path = os.path.join(base_path, "data", "chembl35_data.csv")
df = pd.read_csv(data_path)

# Data Cleaning and Preparation
df = df.dropna(subset=['canonical_smiles', 'standard_value'])  # Drop rows with
missing SMILES or activity values
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce') # Ensure
numeric values
df = df.dropna(subset=['standard_value']) # Remove rows where conversion failed

# RDKit Molecule Conversion and Descriptor Calculation
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    mw = Descriptors.MolWt(mol)
    logp = Chem.Crippen.MolLogP(mol)
    tpsa = Chem.rdMolDescriptors.CalcTPSA(mol)
    return mw, logp, tpsa

df[['MW', 'LogP', 'TPSA']] = df['canonical_smiles'].apply(lambda x:
pd.Series(calculate_descriptors(x)))
df = df.dropna() # Drop rows where descriptor calculation failed

# Data Splitting and Model Training (Simple Linear Regression)
X = df[['MW', 'LogP', 'TPSA']]
y = df['standard_value']  # IC50 values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)
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}")

# Display a few rows of the processed data
print(df.head())
```

Explanation of Python Code:

- **Path Handling:** Uses `os.path.join` to construct the file paths based on your directory structure.
- **Data Loading:** Reads the CSV file created from the SQL query.
- **Data Cleaning:** Handles missing values in 'canonical_smiles' and 'standard_value'. It also converts the 'standard_value' to a numeric type and removes non-numeric values.
- **RDKit Integration:**

    - Converts SMILES strings to RDKit molecule objects.
    - Calculates molecular weight (MW), LogP, and TPSA using RDKit functions.
- **Simple Linear Regression:**
    - Splits the data into training and testing sets.
    - Trains a linear regression model to predict activity based on the calculated descriptors.
    - Evaluates the model using mean squared error (MSE) and R-squared.
- **Error Handling:** The `calculate_descriptors` function includes basic error handling to deal with invalid SMILES strings. This is crucial.
- **Output:** Prints the MSE, R-squared, and the head of the processed dataframe.

Python Code (`notebook/Topic_CheMBL_35_69_2_eda_and_visualization.ipynb`)

```
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns

# Define the base path
base_path = os.path.abspath(os.path.join(os.getcwd(), ".."))

# Data Loading
data_path = os.path.join(base_path, "data", "chembl35_data.csv")
df = pd.read_csv(data_path)

# Basic EDA
print(df.describe())
print(df.info())

# Histograms
plt.figure(figsize=(10, 6))
sns.histplot(df['standard_value'], kde=True)
plt.title('Distribution of IC50 Values')
plt.xlabel('IC50 (nM)')
plt.ylabel('Frequency')
plt.show()

# Scatter Plots (Requires descriptor calculation from previous notebook)
# Assuming you have MW, LogP, TPSA columns
if 'MW' in df.columns and 'LogP' in df.columns and 'TPSA' in df.columns:
    plt.figure(figsize=(12, 8))
    plt.scatter(df['MW'], df['standard_value'])
    plt.title('MW vs IC50')
    plt.xlabel('Molecular Weight')
    plt.ylabel('IC50 (nM)')
    plt.show()

    plt.figure(figsize=(12, 8))
    plt.scatter(df['LogP'], df['standard_value'])
    plt.title('LogP vs IC50')
    plt.xlabel('LogP')
    plt.ylabel('IC50 (nM)')
    plt.show()

    plt.figure(figsize=(12, 8))
    plt.scatter(df['TPSA'], df['standard_value'])
    plt.title('TPSA vs IC50')
    plt.xlabel('TPSA')
    plt.ylabel('IC50 (nM)')
```

```
    plt.show()

# Box Plots (If applicable, for categorical data - e.g., different chembl_ids if
you're working with multiple compounds)
# Example, if you have different compounds (chembl_id):
    plt.figure(figsize=(12, 8))
    sns.boxplot(x='chembl_id', y='standard_value', data=df)
    plt.title('IC50 values for different compounds')
    plt.xticks(rotation=45, ha='right')
    plt.show()


# Correlation Matrix (If descriptors are calculated)
    correlation_matrix = df[['standard_value', 'MW', 'LogP', 'TPSA']].corr()
    sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
    plt.title('Correlation Matrix')
    plt.show()

else:
    print("Descriptors (MW, LogP, TPSA) not found in the dataframe.  Run the data
processing notebook first.")
```

Explanation of EDA Notebook:

- **Loading Data:** Loads the same CSV data as the first notebook.
- **Basic Statistics:** Uses `df.describe()` and `df.info()` to get summary statistics and data types.
- **Histograms:** Shows the distribution of IC50 values.
- **Scatter Plots:** Creates scatter plots of MW, LogP, and TPSA against IC50 values. *Important: These plots will only work correctly if you have already calculated the descriptors in the first notebook.*
- **Box Plots (Example):** Shows how to create box plots to compare IC50 values for different compounds (if applicable to your data). This part is more generic; you might need to adapt it to your specific dataset and research question.
- **Correlation Matrix:** Calculates and visualizes the correlation matrix between activity and descriptors. This part also requires the descriptors to be calculated.
- **Error Handling:** Checks for the existence of the descriptor columns before creating scatter plots.

3. Five Examples (Illustrative)

These are conceptual examples of what you could explore. Given your small dataset, these are primarily demonstrations of *how* to do certain analyses, not definitive scientific findings.

1. **Target-Specific Activity Distribution:** Focus on a single Chembl target ID. Plot a histogram of the IC50 values for compounds active against that target. What is the range of activity? Is it normally distributed?

2. **Descriptor vs. Activity Correlation (One or Two Descriptors):** Calculate MW and LogP. Create a scatter plot of LogP vs. IC50. Calculate the Pearson correlation coefficient between these two variables. *Be cautious about interpreting correlation with only 100 data points!*

3. **Filtering for Highly Potent Compounds:** Filter the dataset to include only compounds with IC50 < 100 nM (highly potent). How many compounds meet this criterion? What are the average MW and LogP values for this subset of compounds?

4. **Comparing Activity to a Known Drug (If Applicable):** If there's a known drug that targets the same protein and is in your dataset, compare its IC50 and descriptors to the rest of the compounds. Where does it fall in the distribution of activity and descriptor space?

5. **Simple Linear Regression (As in the code example, but potentially focusing on a single descriptor):** Train a simple linear regression model to predict IC50 from LogP. Evaluate the model's performance (R-squared). Visualize the regression line on a scatter plot of LogP vs. IC50.

4. Important Considerations in both Language

- **Target Selection is Key:** The SQL query is currently set up to filter for EGFR (CHEMBL205). You *must* change this to the Chembl ID of the target you are interested in analyzing. If you leave it as EGFR, you'll only get data for that target.
- **Units:** Ensure all your activity values are in the same units (nM in this case).
- **Data Quality:** Chembl data can be noisy. Use the confidence score to filter for high-quality data.
- **Small Sample Size:** Remember that your analyses are limited by the small sample size. Be careful about over-interpreting the results. Focus on demonstrating your workflow and identifying potential areas for further investigation.
- **Dependencies:** Make sure you have all the necessary libraries installed (pandas, rdkit, scikit-learn, matplotlib, seaborn). Use `pip install pandas rdkit scikit-learn matplotlib seaborn`.

Important Notes (English)

This revised response provides:

- Corrected SQL code with a focus on target selection, regular expression fix, and a limit clause.
- Python code that integrates with the SQL output, calculates descriptors, and performs simple modeling.
- Clear instructions on how to use `os.path.join` to manage file paths.
- Specific examples of analyses you can perform with your data, given the limitations.
- Emphasis on the importance of target selection, data quality, and the limitations of a small sample size.

Catatan Penting (Bahasa Indonesia)

Respon yang direvisi ini menyediakan:

- Kode SQL yang dikoreksi dengan fokus pada pemilihan target, perbaikan ekspresi reguler, dan klausa pembatas.
- Kode Python yang terintegrasi dengan output SQL, menghitung deskriptor, dan melakukan pemodelan sederhana.
- Instruksi yang jelas tentang cara menggunakan `os.path.join` untuk mengelola jalur file.
- Contoh spesifik analisis yang dapat Anda lakukan dengan data Anda, mengingat keterbatasan yang ada.
- Penekanan pada pentingnya pemilihan target, kualitas data, dan keterbatasan ukuran sampel yang kecil.

Let me know if you have any other questions. Good luck with your project!