PharmApp Suite

🧠 AI for Drug Discovery and Development 🍼

🧩 Topic: Linking ChEMBL with UniProt for protein info

Okay, I understand. You are working on a Chembl 35 data analysis project, using RDKit in a Jupyter Notebook environment structured according to the AIMLops template. You're connecting to a PostgreSQL database and want to perform some data analysis, but you've encountered a few errors. Let's break down the process, address the errors, and create example code snippets.

1. Analysis Model (Análisis del Modelo)

The core idea is to leverage the ChEMBL database (version 35) through SQL queries to extract relevant data. This data is then processed using RDKit in Python for tasks like:

- **Filtering and Cleaning:** Removing irrelevant data based on activity values, ensuring data quality.
- **Structural Representation:** Converting SMILES strings to RDKit Mol objects for feature extraction.
- **Feature Extraction:** Calculating molecular descriptors (e.g., molecular weight, LogP, etc.) using RDKit.
- **Data Analysis/Modeling:** This could involve exploring relationships between molecular descriptors and activity values. Given the error "operator does not exist: numeric ~ unknown", and your target of 100 rows, a simple regression model could be a good starting point. Since your data size is limited, avoid complex models.
- **Visualization:** Plotting distributions of molecular descriptors or scatter plots to visualize correlations.

The AIMLops structure ensures that data, code, and models are organized and easily reproducible.

2. Addressing Errors (Abordando los Errores)

- **Error a:** `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`

  This error arises from trying to use the regular expression operator `~` on a `numeric` column (`act.standard_value`) in PostgreSQL. PostgreSQL doesn't automatically convert `numeric` to `text` for regex matching. There are two solutions:

  - **Cast to TEXT:** Explicitly cast the `standard_value` column to `TEXT` before using the regex operator. `CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'`
  - **Use a numeric comparison:** If you just want to select the values that are positive numbers, using numeric operations might be better. `act.standard_value >= 0`
    Choose the one that best suits your needs and the format of your data.
- **Error b:** `old scikit-learn version does not support parameters squared=False in the mean_squared_error function`

  This indicates you're using an older version of scikit-learn. The `squared=False` parameter was added to `mean_squared_error` to return the Root Mean Squared Error (RMSE) instead of the Mean Squared Error (MSE).

  - **Solution 1: Upgrade scikit-learn:** The recommended solution is to upgrade to a newer version of scikit-learn using `pip install -U scikit-learn`.

- **Solution 2: Calculate RMSE manually:** If upgrading is not feasible, calculate the RMSE manually by taking the square root of the MSE: `rmse = np.sqrt(mean_squared_error(y_true, y_pred))`

3. Code (Código)

Here's the SQL and Python code incorporating the solutions and aiming for 100 rows. I'll use the "cast to TEXT" solution for the SQL error.

SQL (data/<your_file_name>.sql):

```sql
-- Topic_CheMBL_35_81.sql
-- Create table to store chemical and activity data
DROP TABLE IF EXISTS chembl_35_81_activities;

CREATE TABLE chembl_35_81_activities AS
SELECT
    act.molregno,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    md.chembl_id,
    md.smiles
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
WHERE
    act.standard_type = 'IC50'  -- Example activity type
    AND act.standard_units = 'nM'  -- Example activity unit
    AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$' -- Ensure standard_value is a
number
LIMIT 100;

-- Export to CSV (This is done outside the SQL file, using pgAdmin or psql)
-- \copy (SELECT * FROM chembl_35_81_activities) TO
'path/to/your/data/Topic_CheMBL_35_81.csv' DELIMITER ',' CSV HEADER;
```

Explanation:

- `DROP TABLE IF EXISTS chembl_35_81_activities;`: Drops the table if it already exists to avoid errors on repeated runs.
- `CREATE TABLE chembl_35_81_activities AS ...`: Creates a new table named `chembl_35_81_activities` containing the results of the query.
- `SELECT ... FROM activities act JOIN molecule_dictionary md ON act.molregno = md.molregno`: Joins the `activities` and `molecule_dictionary` tables on the `molregno` column, which is the foreign key relating the two tables. This retrieves activity data and molecular information.
- `WHERE ...`: Filters the data based on:
  - `act.standard_type = 'IC50'`: Only selects activities where the `standard_type` is 'IC50'. You can change this to another activity type if needed.
  - `act.standard_units = 'nM'`: Only selects activities where the `standard_units` are 'nM'. Adjust this if needed.
  - `CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'`: Ensures that the `standard_value` is a valid number (only contains digits and a decimal point). It casts the numeric value to TEXT to allow the regex operator to function.
- `LIMIT 100`: Limits the result set to 100 rows.

- The final `--EXPORT` line is a comment; you'd execute that in `psql` or pgAdmin *after* running the `CREATE TABLE` statement to export the data to a CSV file. Replace `'path/to/your/data/Topic_CheMBL_35_81.csv'` with the actual path. Remember to include the `DELIMITER ',' CSV HEADER` part.

Python (notebook/Topic_CheMBL_35_81_1_data_analysis.ipynb):

```
# Topic_CheMBL_35_81_1_data_analysis.ipynb
import os
import pandas as pd
import numpy as np
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
import matplotlib.pyplot as plt

# Define base path
base_path = os.getcwd() # Or specify your AIMLops base path here

# Data Loading
data_path = os.path.join(base_path, "data", "Topic_CheMBL_35_81.csv")
try:
    df = pd.read_csv(data_path)
except FileNotFoundError:
    print(f"Error: File not found at {data_path}.  Make sure you've exported the CSV
from PostgreSQL.")
    exit()

print(f"Data loaded successfully. Shape: {df.shape}")
print(df.head())

# Data Preparation
# Convert IC50 to pIC50 (optional but often useful for activity modeling)
df = df[df['standard_value'].notna()] #Drop NA for proper conversion
df['pIC50'] = -np.log10(df['standard_value'].astype(float) / 1e9) # nM to Molar, then
-log10
print(df.head())

# RDKit Feature Extraction
def calculate_descriptors(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    descriptors = {}
    descriptors['MolWt'] = Descriptors.MolWt(mol)
    descriptors['LogP'] = Descriptors.MolLogP(mol)  #LogP calculation
    descriptors['HBD'] = Descriptors.NumHDonors(mol) #H Bond Donors
    descriptors['HBA'] = Descriptors.NumHAcceptors(mol) #H Bond Acceptors
    return descriptors

df['descriptors'] = df['smiles'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors']) # Drop rows where descriptor calculation failed
df = df[df['descriptors'].apply(lambda x: isinstance(x, dict))] #keep only valid
descriptor results
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1) #Expand the descriptor colums

print(df.head())
```

```
# Data Splitting
X = df[['MolWt', 'LogP', 'HBD', 'HBA']] # Feature selection (adjust as needed)
y = df['pIC50']

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)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)  # Calculate RMSE manually if needed

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

# Visualization
plt.scatter(y_test, y_pred)
plt.xlabel("Actual pIC50")
plt.ylabel("Predicted pIC50")
plt.title("Actual vs. Predicted pIC50")
plt.show()
```

Explanation:

- **Imports:** Imports necessary libraries.
- **`base_path`:** Sets the base path for your AIMLops project. Adjust `os.getcwd()` if your notebook is not in the project's root.
- **Data Loading:** Loads the CSV file you exported from PostgreSQL.
- **Data Preparation:**
  - Converts IC50 values to pIC50 values. This is a common transformation in drug discovery as it distributes the activity data more evenly and makes it easier to model.
- **RDKit Feature Extraction:**
  - Defines a function `calculate_descriptors` that takes a SMILES string as input and calculates molecular descriptors using RDKit.
  - Applies this function to the `smiles` column of your DataFrame to create a new `descriptors` column.
- **Data Splitting:** Splits the data into training and testing sets.
- **Model Training:** Trains a linear regression model. Given your small dataset, a simple model is preferable.
- **Model Evaluation:**
  - Calculates the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) to evaluate the model's performance.
- **Visualization:** Creates a scatter plot to visualize the relationship between actual and predicted pIC50 values.

4. 5 Examples (5 Ejemplos)

Here are five variations you can try, focusing on different aspects of the analysis:

Example 1: Different Activity Type (Tipo de Actividad Diferente)

Change the `standard_type` in the SQL query to a different activity type, such as `'Ki'` or `'EC50'`.

SQL:

```
WHERE
    act.standard_type = 'Ki'  -- Changed to Ki
    AND act.standard_units = 'nM'
    AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'
```

Example 2: Different Descriptors (Descriptores Diferentes)

Add or change the molecular descriptors calculated in the `calculate_descriptors` function in the Python code. For example:

```
descriptors['TPSA'] = Descriptors.TPSA(mol)  # Topological Polar Surface Area
descriptors['NumRotatableBonds'] = Descriptors.NumRotatableBonds(mol)
```

And update the feature selection in your python file:

```
X = df[['MolWt', 'LogP', 'HBD', 'HBA', 'TPSA', 'NumRotatableBonds']] # Feature
selection (adjust as needed)
```

Example 3: Different Regression Model (Modelo de Regresión Diferente)

Try a different regression model, such as Ridge Regression or Lasso Regression, from `sklearn.linear_model`.

```
from sklearn.linear_model import Ridge
model = Ridge(alpha=1.0)  # alpha is a regularization parameter
```

Example 4: Standardization/Scaling (Estandarización/Escalado)

Before training the model, standardize or scale the features using `sklearn.preprocessing`. This can often improve model performance.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Example 5: Explore Different Units (Explorar Diferentes Unidades)

Modify your SQL query to select different units. You'll need to adjust your pIC50 conversion accordingly in the Python code. For example, if you select activities in 'uM' (micromolar):

SQL:

```
WHERE
    act.standard_type = 'IC50'
    AND act.standard_units = 'uM' -- Changed to uM
    AND CAST(act.standard_value AS TEXT) ~ '^[0-9\.]+$'
```

Python (adjust pIC50 calculation):

```
df['pIC50'] = -np.log10(df['standard_value'].astype(float) / 1e6) # uM to Molar, then
-log10
```

Key Considerations:

- **Error Handling:** The provided code includes basic error handling (e.g., checking if the data file exists). You should add more robust error handling to handle potential issues during descriptor calculation or model training.

- **Feature Selection/Engineering:** The choice of molecular descriptors and the feature selection process are critical for building a good model. Experiment with different descriptors and feature selection techniques.
- **Model Validation:** With only 100 rows, your model's generalizability is limited. Consider cross-validation techniques if you can obtain more data.
- **Data Exploration:** Before jumping into modeling, take time to explore the data. Create histograms and scatter plots to understand the distribution of activity values and molecular descriptors.

Remember to adapt the code and examples to your specific needs and the characteristics of your data. Let me know if you have more questions.