# PharmApp Suite

## 🧠 AI for Drug Discovery and Development 🧪

Copyright 2025 | RnD Pharma Plus | www.nghiencuuthuoc.com | Zalo: +84888999311 |

## 🧩 Topic: Mining kinase inhibitors from ChEMBL

Okay, I understand. As a Pharmaceutical Research and Development Specialist, I can help you structure your Chembl 35 data analysis project and address the errors you've encountered. Let's break this down into SQL and Python code, adhering to the AIMLops template and your specific requirements.

### 1. Analysis Model

The project revolves around analyzing ChEMBL 35 data to identify potential drug candidates. Your current workflow involves:

- **Data Extraction (SQL):** Querying the ChEMBL 35 database to retrieve relevant bioactivity data. This involves filtering bioactivities based on specific criteria like target, assay type, and activity type (e.g., IC50). You'll save the results as a CSV file.
- **Data Processing and Analysis (Python):** Using RDKit for molecular processing (e.g., calculating molecular properties, generating fingerprints) and potentially scikit-learn for building predictive models (e.g., Quantitative Structure-Activity Relationship or QSAR models).

### Specific Challenges Addressed:

- **SQL `numeric ~ unknown` error:** This arises when you're trying to use a regular expression (`~` operator) on a numeric column. PostgreSQL is trying to implicitly cast the `standard_value` to text for the regex comparison, and that process fails with numerics. Solution: Explicitly cast `standard_value` to `TEXT` or use numeric comparison.
- **`squared=False` error in scikit-learn:** Your scikit-learn version is outdated. Solution: Either upgrade scikit-learn or use the `mean_squared_error` function without the `squared` parameter (if you don't need the Root Mean Squared Error).

### General workflow consideration

- It's essential to define the specific research question you're trying to answer with this data. For example:
    - "Can we predict the activity (e.g., IC50) of a molecule against a specific target using its molecular properties?"
    - "What are the key molecular features that correlate with high activity against a particular target?"

  Having a clear research question will guide your feature selection, model choice, and evaluation metrics.

### 2. Code Examples (SQL and Python)

Let's assume your `base_path` is defined as follows:

```python
import os

base_path = "./" # Or a more specific path, such as "/path/to/your/project"
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")
os.makedirs(data_path, exist_ok=True)  # Create the data directory if it doesn't exist
os.makedirs(notebook_path, exist_ok=True) # Create the notebooks directory if it doesn't exist
```

**SQL Code (Topic_CheMBL_35_17.sql - saved in `../data/` folder via pgAdmin)**

```
-- Connect to the chembl_35 database
\c chembl_35

-- Extract bioactivity data for a specific target (example: Target ID CHEMBL205 -
Tyrosine-protein kinase ABL1)
-- Limiting to 100 rows for demonstration purposes

SELECT
    act.molregno,
    act.standard_value,
    act.standard_units,
    act.standard_type,
    md.chembl_id AS molecule_chembl_id,
    td.chembl_id AS target_chembl_id
FROM
    activities act
JOIN
    molecule_dictionary md ON act.molregno = md.molregno
JOIN
    target_dictionary td ON act.tid = td.tid
WHERE
    td.chembl_id = 'CHEMBL205'  -- Example: Target ID for Tyrosine-protein kinase ABL1
    AND act.standard_type = 'IC50' -- Filter for IC50 values
    AND act.standard_units = 'nM'  -- Filter for nM units
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0      --Avoid zero values

    --Explicitly cast standard_value to TEXT to avoid the numeric ~ unknown error.
alternative we can use act.standard_value >= 0
    AND act.standard_value::TEXT ~ '^[0-9\.]+$'

ORDER BY act.standard_value ASC
LIMIT 100;

-- Save the results to a CSV file (you'll do this manually through pgAdmin)
-- Export the result of the query as CSV from pgAdmin directly into data folder
```

**Important:** After running the SQL query, use pgAdmin's export feature to save the results as `Topic_CheMBL_35_17.csv` in the data directory ( `../data/Topic_CheMBL_35_17.csv` ).

**Python Code (Topic_CheMBL_35_17_1_Data_Loading_and_Preprocessing.ipynb - saved in notebooks folder)**

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

# Define paths
base_path = "./" # Or a more specific path, such as "/path/to/your/project"
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")
csv_file = os.path.join(data_path, "Topic_CheMBL_35_17.csv")


# Load the data
try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}.  Make sure you have saved the SQL
query output as a CSV file in the data folder.")
    exit()

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

# Basic data cleaning and preparation
df = df.dropna(subset=['molregno', 'standard_value']) # Remove rows with missing
values
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce') # Ensure
standard_value is numeric
df = df[df['standard_value'] > 0] #Remove values equal or less than zero
df = df.drop_duplicates(subset=['molregno']) #Remove duplicated rows, by molregno
print(f"Data cleaned. Shape: {df.shape}")

# Function to calculate molecular weight using RDKit
def calculate_mw(molregno):
    try:
        # Fetch the SMILES string from ChEMBL (you might need another query or a pre-
existing table)
        # This is just a placeholder - replace with your actual method for getting
SMILES
        # For example, you could load a separate CSV with molregno and SMILES
        # In this example, just for demonstration, let's assume you have SMILES
available
        # and you're adding it as a new column from another dataframe named smiles_df
        # smiles = smiles_df[smiles_df['molregno'] == molregno]['smiles'].iloc[0]
        #  IF smiles_df not defined, make a fake smile to let the code run
        smiles = "CC(=O)Oc1ccccc1C(=O)O" # Replace with actual SMILES retrieval
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            return Descriptors.MolWt(mol)
        else:
            return None
    except Exception as e:
        print(f"Error processing molregno {molregno}: {e}")
        return None

# Apply the function to calculate molecular weight
df['molecular_weight'] = df['molregno'].apply(calculate_mw)
df = df.dropna(subset=['molecular_weight']) #Remove rows with missing values after
calculation

print(df.head())
print(df.dtypes)
```

**Python Code (Topic_CheMBL_35_17_2_QSAR_Model.ipynb - saved in notebooks folder)**

```
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Descriptors
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
```

```python
import pickle

# Define paths
base_path = "./" # Or a more specific path, such as "/path/to/your/project"
data_path = os.path.join(base_path, "data")
notebook_path = os.path.join(base_path, "notebooks")
model_path = os.path.join(base_path, "models")  # New: Path to save models
os.makedirs(model_path, exist_ok=True) # Create models directory
csv_file = os.path.join(data_path, "Topic_CheMBL_35_17.csv")


# Load the data
try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}.  Make sure you have saved the SQL
query output as a CSV file in the data folder.")
    exit()

# Basic data cleaning and preparation
df = df.dropna(subset=['molregno', 'standard_value']) # Remove rows with missing
values
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce') # Ensure
standard_value is numeric
df = df[df['standard_value'] > 0] #Remove values equal or less than zero
df = df.drop_duplicates(subset=['molregno']) #Remove duplicated rows, by molregno

# Function to calculate molecular fingerprints using RDKit
def calculate_fingerprint(molregno):
    try:
        # Fetch the SMILES string from ChEMBL (you might need another query or a pre-
existing table)
        smiles = "CC(=O)Oc1ccccc1C(=O)O" # Replace with actual SMILES retrieval
        mol = Chem.MolFromSmiles(smiles)
        if mol:
            fp = AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=2048)  # Morgan
fingerprint
            return np.array(fp)
        else:
            return None
    except Exception as e:
        print(f"Error processing molregno {molregno}: {e}")
        return None

# Apply the function to calculate fingerprints
df['fingerprint'] = df['molregno'].apply(calculate_fingerprint)
df = df.dropna(subset=['fingerprint'])

# Prepare data for modeling
X = np.vstack(df['fingerprint'].values) #Stacking fingerprints
y = -np.log10(df['standard_value'].values)  # Convert IC50 to pIC50 (more suitable for
modeling)

# 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
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^2 Score: {r2}")

# Save the model
model_filename = os.path.join(model_path, "chembl_qsar_model.pkl")
pickle.dump(model, open(model_filename, 'wb'))
print(f"Model saved to {model_filename}")
```

**3. Five Examples (Use Cases)**

Here are five examples of how you might use this analysis pipeline:

1. **Target Prioritization:** Identify targets with a large number of high-affinity compounds (e.g., low IC50 values). This can help prioritize targets for further investigation.
    - **SQL:** Modify the SQL query to group by `target_chembl_id` and count the number of activities below a certain `standard_value`.
2. **Lead Discovery:** Screen a virtual library of compounds against a specific target and predict their activity using the QSAR model.
    - **Python:** Load the trained QSAR model and use it to predict the activity of new compounds based on their calculated fingerprints.
3. **Hit-to-Lead Optimization:** Identify structural modifications to improve the activity of a lead compound.
    - **Python:** Generate a series of analogs (using RDKit's scaffold hopping or fragment replacement methods) and predict their activity using the QSAR model.
4. **Off-Target Prediction:** Predict the activity of a compound against a panel of targets to identify potential off-target effects.
    - **SQL & Python:** Extend the SQL query to retrieve data for multiple targets. Train separate QSAR models for each target and use them to predict the activity of a compound against the entire panel.
5. **QSAR Model Improvement:** Evaluate the QSAR model performance, refine the features, and try different machine-learning algorithms.
    - **Python:** Experiment with different molecular descriptors (e.g., constitutional, topological, electronic) and machine-learning algorithms (e.g., Support Vector Machines, Neural Networks). Use cross-validation to evaluate the model's performance and hyperparameter optimization techniques (e.g., GridSearchCV) to find the best model configuration.

**Important Considerations:**

- **SMILES retrieval:** The provided code includes placeholders for fetching SMILES strings. You'll need to implement the appropriate method for retrieving SMILES from the ChEMBL database or another source. Consider creating a separate table that links `molregno` to SMILES.
- **Feature Selection:** Experiment with different molecular descriptors and feature selection techniques to improve the QSAR model's performance.

- **Model Validation:** Rigorous model validation is crucial. Use techniques like cross-validation, y-randomization, and external validation to ensure the model's reliability and predictive power.
- **Data scaling:** It might be beneficial to scale features before training the model. This can improve the performance of some machine learning algorithms.

**In summary (English and Chinese):**

**English:** This structured approach, incorporating SQL for data extraction and Python with RDKit for analysis, will allow you to efficiently work with ChEMBL 35 data. Addressing the specific errors mentioned and focusing on a clear research question are key to a successful project. Remember to thoroughly validate your QSAR models.

**Chinese (中文):** 这种结构化的方法，结合了 SQL 用于数据提取和 Python 与 RDKit 用于分析，将使您能够有效地处理 ChEMBL 35 数据。 解决所提到的特定错误并专注于明确的研究问题是项目成功的关键。 记住要彻底验证您的 QSAR 模型。