PharmApp Suite

🧠 AI for Drug Discovery and Development 🧪

🧩 Topic: Predicting binding modes via molecular docking

Okay, I understand. You're working on a Chembl 35 data analysis project using RDKit and want guidance on structuring your code according to an AIMLops template and running your analysis. You're facing issues with SQL and scikit-learn versions. You also need guidance on how to effectively structure your analysis in a modular and reproducible way, and need specific examples to get started.

Here's a breakdown of how to approach this, along with code examples and analysis.

1. Analysis Model and Structure

Your analysis can be structured into the following steps:

- **Data Extraction:** Use SQL to query the Chembl 35 database, extracting the necessary data (activities, compounds, etc.) and saving it as CSV files. Specifically, we'll focus on activity data, filtering for specific activity types (e.g., IC50) and targets.
- **Data Preprocessing:** Load the CSV files into a pandas DataFrame. Clean and prepare the data for analysis. This includes handling missing values, converting data types, and potentially standardizing or normalizing numerical features.
- **Feature Generation:** Use RDKit to generate molecular descriptors from the SMILES strings in your data. These descriptors will serve as features for your models.
- **Model Training and Evaluation:** Choose a relevant machine learning model (e.g., linear regression, random forest) to predict activity based on the generated descriptors. Split your data into training and testing sets, train the model, and evaluate its performance using appropriate metrics (e.g., mean squared error, R-squared).
- **Analysis and Interpretation:** Analyze the model's performance, identify important features, and draw conclusions about the structure-activity relationship.

2. Addressing the Errors

- **Error 4.a:** `ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`

  This error occurs because you are trying to use a regular expression (~) on a numeric column (`act.standard_value`). The regular expression operator in PostgreSQL is primarily used for text matching. To fix this, you should use appropriate numeric comparison operators. If you need to filter based on specific numerical patterns, you might need to cast the column to text first, but it's generally better to use numerical comparisons.

- **Error 4.b:** `old scikit-learn version does not support parameters squared=False in the mean_squared_error function`

  This indicates that you are using an older version of scikit-learn. The `squared=False` parameter was introduced in a later version to return the root mean squared error (RMSE) directly. To fix this, you have two options:

  1. **Upgrade scikit-learn:** This is the recommended approach. Use `pip install -U scikit-learn` to upgrade to the latest version.
  2. **Calculate RMSE manually:** If you cannot upgrade scikit-learn, calculate the RMSE by taking the square root of the mean squared error: `rmse = np.sqrt(mean_squared_error(y_true, y_pred))`. You'll need to import `numpy` as `np`.

3. AIMLops Folder Structure and Code Organization

Following the AIMLops template, your project structure could look like this:

```
Project_Root/
├── data/                  # CSV files extracted from Chembl
├── notebooks/             # Jupyter notebooks
│   ├── Topic_CheMBL_35_32_1_data_extraction_preprocessing.ipynb
│   ├── Topic_CheMBL_35_32_2_feature_generation.ipynb
│   ├── Topic_CheMBL_35_32_3_model_training_evaluation.ipynb
│   ├── Topic_CheMBL_35_32_4_analysis_interpretation.ipynb
│   └── Topic_CheMBL_35_32_5_example_analysis.ipynb
├── src/                   # Python modules (optional - for reusable code)
├── models/                # Trained models (e.g., pickled files)
├── reports/               # Reports and visualizations
├── requirements.txt       # Python dependencies
└── README.md              # Project documentation
```

4. Code Examples (SQL and Python)

Here are five examples, broken down by notebook purpose:

Example 1: `Topic_CheMBL_35_32_1_data_extraction_preprocessing.ipynb` - Data Extraction and Preprocessing

SQL Code (to run on pgAdmin and save to `../data/chembl_activity_data.csv`)

```sql
-- Select data from chembl_35.activities and related tables
SELECT
    act.activity_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    act.relation,
    cmp.molecule_structures,
    t.tid,
    t.pref_name
FROM
    chembl_35.activities act
JOIN
    chembl_35.assays ass ON act.assay_id = ass.assay_id
JOIN
    chembl_35.target_dictionary t ON ass.tid = t.tid
JOIN
    chembl_35.compound_structures cmp ON act.molregno = cmp.molregno
WHERE
    act.standard_type = 'IC50' -- Focus on IC50 values
    AND act.standard_units = 'nM' -- Ensure units are in nM
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0  -- Exclude non-positive values
    AND t.pref_name = 'Acetylcholinesterase'
LIMIT 100; -- Limit to 100 rows
```

Python Code (within `Topic_CheMBL_35_32_1_data_extraction_preprocessing.ipynb`)

```python
import os
import pandas as pd

base_path = os.getcwd()  # Current directory (Project_Root)
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.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've run the SQL and
saved the CSV.")
    exit()

print(f"Shape of the dataframe: {df.shape}")
print(df.head())

# Data cleaning (example - handling missing values)
df = df.dropna(subset=['standard_value', 'molecule_structures'])

# Convert standard_value to numeric
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')  # Convert
to numeric

print(f"Shape of the dataframe after removing NA: {df.shape}")
print(df.dtypes)
```

Example 2: `Topic_CheMBL_35_32_2_feature_generation.ipynb` - Feature Generation

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

base_path = os.getcwd()
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}. Make sure you've run the previous
notebook and saved the CSV.")
    exit()

#Data Cleaning

df = df.dropna(subset=['standard_value', 'molecule_structures'])
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')  # Convert
to numeric

# Function to calculate molecular descriptors
def calculate_descriptors(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None  # Handle invalid SMILES strings
        descriptors = {}
        descriptors['MolWt'] = Descriptors.MolWt(mol)
        descriptors['LogP'] = Descriptors.MolLogP(mol)
        descriptors['HBD'] = Descriptors.NumHDonors(mol)
        descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
        return descriptors
    except Exception as e:
        print(f"Error processing SMILES: {smiles} - {e}")
        return None

# Apply the function to create new columns
df['descriptors'] = df['molecule_structures'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors'])  # Remove rows where descriptor calculation
failed

# Expand the 'descriptors' column into separate columns
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

print(df.head())
```

Example 3: `Topic_CheMBL_35_32_3_model_training_evaluation.ipynb` - Model Training and Evaluation

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

base_path = os.getcwd()
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}. Make sure you've run the previous
notebook and saved the CSV.")
    exit()

# Drop rows with missing values in relevant columns
df = df.dropna(subset=['MolWt', 'LogP', 'HBD', 'HBA', 'standard_value'])

# Define features (X) and target (y)
X = df[['MolWt', 'LogP', 'HBD', 'HBA']]
y = df['standard_value']

# 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 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)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R-squared: {r2}")
```

Example 4: `Topic_CheMBL_35_32_4_analysis_interpretation.ipynb` - Analysis and Interpretation

```python
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns

base_path = os.getcwd()
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}. Make sure you've run the previous
notebook and saved the CSV.")
    exit()

# Drop rows with missing values in relevant columns
df = df.dropna(subset=['MolWt', 'LogP', 'HBD', 'HBA', 'standard_value'])

# Define features (X) and target (y)
X = df[['MolWt', 'LogP', 'HBD', 'HBA']]
y = df['standard_value']

# 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 linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Get the coefficients
coefficients = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_})
coefficients = coefficients.sort_values(by='Coefficient', ascending=False)

print("Model Coefficients:")
print(coefficients)

# Visualize predicted vs. actual values
y_pred = model.predict(X_test)
plt.figure(figsize=(8, 6))
sns.regplot(x=y_test, y=y_pred, scatter_kws={'alpha':0.5})
plt.xlabel("Actual IC50 (nM)")
plt.ylabel("Predicted IC50 (nM)")
plt.title("Actual vs. Predicted IC50 Values")
plt.show()
```

Example 5: `Topic_CheMBL_35_32_5_example_analysis.ipynb` - Example Specific Analysis (e.g., Activity Cliff Detection)

```python
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Descriptors
from rdkit.Chem import MACCSkeys
from rdkit.DataStructs import FingerprintSimilarity
import numpy as np

base_path = os.getcwd()
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File not found at {csv_file}. Make sure you've run the previous
notebook and saved the CSV.")
    exit()

#Data Cleaning

df = df.dropna(subset=['standard_value', 'molecule_structures'])
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')  # Convert
to numeric

# Calculate MACCS keys fingerprints
def calculate_maccs_keys(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None
        return MACCSkeys.GenMACCSKeys(mol)
    except:
        return None

df['maccs_keys'] = df['molecule_structures'].apply(calculate_maccs_keys)
df = df.dropna(subset=['maccs_keys'])

# Function to calculate similarity
def calculate_similarity(fp1, fp2):
    return FingerprintSimilarity(fp1, fp2)

# Select two compounds
compound1 = df.iloc[0]
compound2 = df.iloc[1]

# Calculate similarity
similarity = calculate_similarity(compound1['maccs_keys'], compound2['maccs_keys'])
print(f"Similarity between compound 1 and compound 2: {similarity}")
print(f"Activity of Compound 1: {compound1['standard_value']}")
print(f"Activity of Compound 2: {compound2['standard_value']}")

activity_difference = abs(compound1['standard_value'] - compound2['standard_value'])
print(f"Activity Difference: {activity_difference}")

#Determine Activity Cliff
if similarity < 0.8 and activity_difference > 10:  #Adjust Threshold as needed
    print("This might be an activity cliff!")
else:
    print("Not likely an activity cliff.")
```

Explanation:

- **Modularization:** The code is broken down into logical notebooks, making it easier to understand, maintain, and debug.
- **File Handling:** The `os.path.join` function is used consistently for file paths, ensuring portability across different operating systems.
- **Error Handling:** The code includes `try...except` blocks to handle potential errors during file loading and descriptor calculation.
- **Comments:** The code is well-commented to explain the purpose of each section.
- **Dependencies:** A `requirements.txt` file should be created to list all the Python dependencies used in the project (e.g., `pandas`, `rdkit`, `scikit-learn`). You can generate this file using `pip freeze > requirements.txt`.
- **Version Control:** Use Git to track changes to your code and collaborate with others.
- **Data Validation:** Include data validation steps to ensure that the data meets your expectations (e.g., checking for missing values, data types, and valid ranges).
- **Visualization:** Use libraries like Matplotlib and Seaborn to create informative visualizations of your data and model results.
- **Logging:** Consider using a logging library to record important events and errors during the execution of your code.

Instructions:

1. **Install Dependencies:** Create a `requirements.txt` file with the following content (or add to your existing file):

   ```
   pandas
   rdkit
   scikit-learn
   matplotlib
   seaborn
   numpy
   ```

   Then, run `pip install -r requirements.txt` to install all the necessary packages.

2. **Create Folders:** Create the `data`, `notebooks`, `src`, `models`, and `reports` folders in your project directory.

3. **SQL Execution:** Run the SQL query on your pgAdmin connected to the specified server, user, password and database.

4. **Run Notebooks:** Execute the code in each Jupyter notebook sequentially. Make sure that each notebook can access the data generated by the previous notebook.

5. **Adjust Code:** Adjust the code to fit your specific needs and the structure of your data. Experiment with different models, features, and evaluation metrics.

Important Considerations:

- **CheMBL License:** Make sure you comply with the CheMBL data license.
- **Data Privacy:** Be mindful of data privacy and security when working with sensitive data.
- **Reproducibility:** Strive for reproducibility by using a consistent environment, version control, and well-documented code.
- **Model Selection:** The choice of machine learning model depends on the specific problem you are trying to solve and the characteristics of your data. Experiment with different models to find the best one for your needs.
- **Feature Engineering:** Feature engineering is a crucial step in machine learning. Experiment with different molecular descriptors and combinations of features to improve the performance of your models.

- **Validation:** Thoroughly validate your models using appropriate validation techniques (e.g., cross-validation) to ensure that they generalize well to unseen data.

Let me know if you have any specific questions or need help with any of these steps. Good luck with your project!

Indonesian Translation

Oke, saya mengerti. Anda sedang mengerjakan proyek analisis data Chembl 35 menggunakan RDKit dan memerlukan panduan tentang bagaimana menstrukturkan kode Anda sesuai dengan template AIMLops dan menjalankan analisis Anda. Anda menghadapi masalah dengan SQL dan versi scikit-learn. Anda juga memerlukan panduan tentang bagaimana menstrukturkan analisis Anda secara efektif dalam cara yang modular dan dapat direproduksi, dan memerlukan contoh spesifik untuk memulai.

Berikut adalah uraian tentang bagaimana mendekati hal ini, beserta contoh kode dan analisis.

1. Model dan Struktur Analisis

Analisis Anda dapat distrukturkan ke dalam langkah-langkah berikut:

- **Ekstraksi Data:** Gunakan SQL untuk menanyakan database Chembl 35, mengekstrak data yang diperlukan (aktivitas, senyawa, dll.) dan menyimpannya sebagai file CSV. Secara khusus, kita akan fokus pada data aktivitas, memfilter jenis aktivitas tertentu (misalnya, IC50) dan target.
- **Praproses Data:** Muat file CSV ke dalam DataFrame pandas. Bersihkan dan siapkan data untuk analisis. Ini termasuk menangani nilai yang hilang, mengonversi tipe data, dan berpotensi menstandardisasi atau menormalkan fitur numerik.
- **Pembuatan Fitur:** Gunakan RDKit untuk menghasilkan deskriptor molekul dari string SMILES dalam data Anda. Deskriptor ini akan berfungsi sebagai fitur untuk model Anda.
- **Pelatihan dan Evaluasi Model:** Pilih model pembelajaran mesin yang relevan (misalnya, regresi linier, random forest) untuk memprediksi aktivitas berdasarkan deskriptor yang dihasilkan. Pisahkan data Anda menjadi set pelatihan dan pengujian, latih model, dan evaluasi kinerjanya menggunakan metrik yang sesuai (misalnya, mean squared error, R-squared).
- **Analisis dan Interpretasi:** Analisis kinerja model, identifikasi fitur penting, dan tarik kesimpulan tentang hubungan struktur-aktivitas.

2. Mengatasi Kesalahan

- **Kesalahan 4.a:** **`ERROR: operator does not exist: numeric ~ unknown, LINE 12: AND act.standard_value ~ '^[0-9\.]+$'`**

  Kesalahan ini terjadi karena Anda mencoba menggunakan ekspresi reguler (~) pada kolom numerik (`act.standard_value`). Operator ekspresi reguler di PostgreSQL terutama digunakan untuk pencocokan teks. Untuk memperbaiki ini, Anda harus menggunakan operator perbandingan numerik yang sesuai. Jika Anda perlu memfilter berdasarkan pola numerik tertentu, Anda mungkin perlu mentransmisikan kolom ke teks terlebih dahulu, tetapi umumnya lebih baik menggunakan perbandingan numerik.

- **Kesalahan 4.b:** **`old scikit-learn version does not support parameters squared=False in the mean_squared_error function`**

  Ini menunjukkan bahwa Anda menggunakan versi scikit-learn yang lebih lama. Parameter `squared=False` diperkenalkan dalam versi yang lebih baru untuk mengembalikan root mean squared error (RMSE) secara langsung. Untuk memperbaiki ini, Anda memiliki dua opsi:

  1. **Tingkatkan scikit-learn:** Ini adalah pendekatan yang disarankan. Gunakan `pip install -U scikit-learn` untuk meningkatkan ke versi terbaru.

2. **Hitung RMSE secara manual:** Jika Anda tidak dapat meningkatkan scikit-learn, hitung RMSE dengan mengambil akar kuadrat dari mean squared error: `rmse = np.sqrt(mean_squared_error(y_true, y_pred))`. Anda perlu mengimpor `numpy` sebagai `np`.

3. Struktur Folder AIMLops dan Organisasi Kode

Mengikuti template AIMLops, struktur proyek Anda dapat terlihat seperti ini:

```
Project_Root/
├── data/                  # File CSV yang diekstrak dari Chembl
├── notebooks/             # Jupyter notebooks
│   ├── Topic_CheMBL_35_32_1_data_extraction_preprocessing.ipynb
│   ├── Topic_CheMBL_35_32_2_feature_generation.ipynb
│   ├── Topic_CheMBL_35_32_3_model_training_evaluation.ipynb
│   ├── Topic_CheMBL_35_32_4_analysis_interpretation.ipynb
│   └── Topic_CheMBL_35_32_5_example_analysis.ipynb
├── src/                   # Modul Python (opsional - untuk kode yang dapat digunakan
kembali)
├── models/                # Model terlatih (misalnya, file pickled)
├── reports/               # Laporan dan visualisasi
├── requirements.txt       # Dependensi Python
└── README.md              # Dokumentasi proyek
```

4. Contoh Kode (SQL dan Python)

Berikut adalah lima contoh, dipecah berdasarkan tujuan notebook:

Contoh 1: `Topic_CheMBL_35_32_1_data_extraction_preprocessing.ipynb` - Ekstraksi dan Praproses Data

Kode SQL (untuk dijalankan di pgAdmin dan disimpan ke `../data/chembl_activity_data.csv`)

```sql
-- Pilih data dari chembl_35.activities dan tabel terkait
SELECT
    act.activity_id,
    act.standard_type,
    act.standard_value,
    act.standard_units,
    act.relation,
    cmp.molecule_structures,
    t.tid,
    t.pref_name
FROM
    chembl_35.activities act
JOIN
    chembl_35.assays ass ON act.assay_id = ass.assay_id
JOIN
    chembl_35.target_dictionary t ON ass.tid = t.tid
JOIN
    chembl_35.compound_structures cmp ON act.molregno = cmp.molregno
WHERE
    act.standard_type = 'IC50' -- Fokus pada nilai IC50
    AND act.standard_units = 'nM' -- Pastikan unit dalam nM
    AND act.standard_value IS NOT NULL
    AND act.standard_value > 0  -- Kecualikan nilai non-positif
    AND t.pref_name = 'Acetylcholinesterase'
LIMIT 100; -- Batasi hingga 100 baris
```

Kode Python (di dalam `Topic_CheMBL_35_32_1_data_extraction_preprocessing.ipynb`)

```python
import os
import pandas as pd

base_path = os.getcwd()  # Direktori saat ini (Project_Root)
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

# Muat data
try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File tidak ditemukan di {csv_file}. Pastikan Anda telah menjalankan
SQL dan menyimpan CSV.")
    exit()

print(f"Bentuk dataframe: {df.shape}")
print(df.head())

# Pembersihan data (contoh - menangani nilai yang hilang)
df = df.dropna(subset=['standard_value', 'molecule_structures'])

# Konversi standard_value ke numerik
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')  #
Konversi ke numerik

print(f"Bentuk dataframe setelah menghapus NA: {df.shape}")
print(df.dtypes)
```

Contoh 2: `Topic_CheMBL_35_32_2_feature_generation.ipynb` - Pembuatan Fitur

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

base_path = os.getcwd()
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File tidak ditemukan di {csv_file}. Pastikan Anda telah menjalankan
notebook sebelumnya dan menyimpan CSV.")
    exit()

#Data Cleaning

df = df.dropna(subset=['standard_value', 'molecule_structures'])
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')  #
Konversi ke numerik

# Fungsi untuk menghitung deskriptor molekul
def calculate_descriptors(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None  # Tangani string SMILES yang tidak valid
        descriptors = {}
        descriptors['MolWt'] = Descriptors.MolWt(mol)
        descriptors['LogP'] = Descriptors.MolLogP(mol)
        descriptors['HBD'] = Descriptors.NumHDonors(mol)
        descriptors['HBA'] = Descriptors.NumHAcceptors(mol)
        return descriptors
    except Exception as e:
        print(f"Error memproses SMILES: {smiles} - {e}")
        return None

# Terapkan fungsi untuk membuat kolom baru
df['descriptors'] = df['molecule_structures'].apply(calculate_descriptors)
df = df.dropna(subset=['descriptors'])  # Hapus baris di mana perhitungan deskriptor
gagal

# Perluas kolom 'descriptors' menjadi kolom terpisah
df = pd.concat([df.drop(['descriptors'], axis=1), df['descriptors'].apply(pd.Series)],
axis=1)

print(df.head())
```

Contoh 3: `Topic_CheMBL_35_32_3_model_training_evaluation.ipynb` - Pelatihan dan Evaluasi Model

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

base_path = os.getcwd()
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File tidak ditemukan di {csv_file}. Pastikan Anda telah menjalankan
notebook sebelumnya dan menyimpan CSV.")
    exit()

# Hapus baris dengan nilai yang hilang di kolom yang relevan
df = df.dropna(subset=['MolWt', 'LogP', 'HBD', 'HBA', 'standard_value'])

# Definisikan fitur (X) dan target (y)
X = df[['MolWt', 'LogP', 'HBD', 'HBA']]
y = df['standard_value']

# Pisahkan data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Latih model regresi linier
model = LinearRegression()
model.fit(X_train, y_train)

# Buat prediksi pada set pengujian
y_pred = model.predict(X_test)

# Evaluasi model
```

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R-squared: {r2}")
```

Contoh 4: `Topic_CheMBL_35_32_4_analysis_interpretation.ipynb` - Analisis dan Interpretasi

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns

base_path = os.getcwd()
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File tidak ditemukan di {csv_file}. Pastikan Anda telah menjalankan
notebook sebelumnya dan menyimpan CSV.")
    exit()

# Hapus baris dengan nilai yang hilang di kolom yang relevan
df = df.dropna(subset=['MolWt', 'LogP', 'HBD', 'HBA', 'standard_value'])

# Definisikan fitur (X) dan target (y)
X = df[['MolWt', 'LogP', 'HBD', 'HBA']]
y = df['standard_value']

# Pisahkan data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Latih model regresi linier
model = LinearRegression()
model.fit(X_train, y_train)

# Dapatkan koefisien
coefficients = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_})
coefficients = coefficients.sort_values(by='Coefficient', ascending=False)

print("Koefisien Model:")
print(coefficients)

# Visualisasikan nilai prediksi vs. aktual
y_pred = model.predict(X_test)
plt.figure(figsize=(8, 6))
sns.regplot(x=y_test, y=y_pred, scatter_kws={'alpha':0.5})
plt.xlabel("IC50 Aktual (nM)")
plt.ylabel("IC50 Prediksi (nM)")
plt.title("Nilai IC50 Aktual vs. Prediksi")
plt.show()
```

Contoh 5: `Topic_CheMBL_35_32_5_example_analysis.ipynb` - Contoh Analisis Spesifik (misalnya, Deteksi Activity Cliff)

```python
import os
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Descriptors
from rdkit.Chem import MACCSkeys
from rdkit.DataStructs import FingerprintSimilarity
import numpy as np

base_path = os.getcwd()
data_path = os.path.join(base_path, 'data')
csv_file = os.path.join(data_path, 'chembl_activity_data.csv')

try:
    df = pd.read_csv(csv_file)
except FileNotFoundError:
    print(f"Error: File tidak ditemukan di {csv_file}. Pastikan Anda telah menjalankan
notebook sebelumnya dan menyimpan CSV.")
    exit()

#Data Cleaning

df = df.dropna(subset=['standard_value', 'molecule_structures'])
df['standard_value'] = pd.to_numeric(df['standard_value'], errors='coerce')  #
Konversi ke numerik

# Calculate MACCS keys fingerprints
def calculate_maccs_keys(smiles):
    try:
        mol = Chem.MolFromSmiles(smiles)
        if mol is None:
            return None
        return MACCSkeys.GenMACCSKeys(mol)
    except:
        return None

df['maccs_keys'] = df['molecule_structures'].apply(calculate_maccs_keys)
df = df.dropna(subset=['maccs_keys'])

# Function to calculate similarity
def calculate_similarity(fp1, fp2):
    return FingerprintSimilarity(fp1, fp2)

# Select two compounds
compound1 = df.iloc[0]
compound2 = df.iloc[1]

# Calculate similarity
similarity = calculate_similarity(compound1['maccs_keys'], compound2['maccs_keys'])
print(f"Kesamaan antara senyawa 1 dan senyawa 2: {similarity}")
print(f"Aktivitas Senyawa 1: {compound1['standard_value']}")
print(f"Aktivitas Senyawa 2: {compound2['standard_value']}")

activity_difference = abs(compound1['standard_value'] - compound2['standard_value'])
print(f"Perbedaan Aktivitas: {activity_difference}")
```

```
#Determine Activity Cliff
if similarity < 0.8 and activity_difference > 10:  #Adjust Threshold as needed
    print("Ini mungkin activity cliff!")
else:
    print("Kemungkinan bukan activity cliff.")
```

Penjelasan:

- **Modularisasi:** Kode dipecah menjadi notebook logis, sehingga lebih mudah dipahami, dipelihara, dan di-debug.
- **Penanganan File:** Fungsi `os.path.join` digunakan secara konsisten untuk jalur file, memastikan portabilitas di berbagai sistem operasi.
- **Penanganan Kesalahan:** Kode menyertakan blok `try...except` untuk menangani potensi kesalahan selama pemuatan file dan perhitungan deskriptor.
- **Komentar:** Kode dik