In [17]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressc
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import StackingRegressor, VotingRegressor

# Load your water quality dataset
# Replace 'your_data.csv' with the actual filename
data = pd.read_csv('water_potability.csv')

# Split the data into features (X) and target variable (y)
X = data.drop(columns=['Potability'])
y = data['Potability']
from sklearn.impute import SimpleImputer

# Impute missing values in X
imputer = SimpleImputer(strategy='mean')  # You can choose a different strat
X = imputer.fit_transform(X)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

# Define your base models
base_models = [
    ('random_forest', RandomForestRegressor(n_estimators=100, random_state=4
    ('gradient_boosting', GradientBoostingRegressor(n_estimators=100, random
    ('linear_regression', LinearRegression())
]

# Create a StackingRegressor
stacking_regressor = StackingRegressor(estimators=base_models, final_estimat

# Fit the stacking model on the training data
stacking_regressor.fit(X_train, y_train)

# Make predictions using the stacking model
stacking_predictions = stacking_regressor.predict(X_test)

# Calculate the Mean Squared Error for stacking
stacking_mse = mean_squared_error(y_test, stacking_predictions)
print(f"Stacking Mean Squared Error: {stacking_mse}")
# Define your base models
base_models = [
    ('random_forest', RandomForestRegressor(n_estimators=100, random_state=4
    ('gradient_boosting', GradientBoostingRegressor(n_estimators=100, random
    ('linear_regression', LinearRegression())
]

# Create a StackingRegressor
stacking_regressor = StackingRegressor(estimators=base_models, final_estimat

# Fit the stacking model on the training data
stacking_regressor.fit(X_train, y_train)

# Fit the GradientBoostingRegressor model
base_models[1][1].fit(X_train, y_train)

# Generate predictions for Gradient Boosting and Linear Regression models
gradient_boosting_predictions = base_models[1][1].predict(X_test)
linear_regression_predictions = base_models[1][1].predict(X_test)
```

```
# Make predictions using the stacking model
stacking_predictions = stacking_regressor.predict(X_test)

# Calculate the Mean Squared Error for weighted averaging
weighted_predictions = (0.4 * stacking_predictions + 0.4 * gradient_boosting

# Calculate the Mean Squared Error for weighted averaging
weighted_mse = mean_squared_error(y_test, weighted_predictions)
print(f"Weighted Averaging Mean Squared Error: {weighted_mse}")
```

```
Stacking Mean Squared Error: 0.2061095791393775
Weighted Averaging Mean Squared Error: 0.2079379157348387
```

In [ ]: