import pandas as pd

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.model\_selection import train\_test\_split

import chardet

import matplotlib.pyplot as plt

from matplotlib import font\_manager

# Set font to support Chinese characters

plt.rcParams['font.sans-serif'] = ['SimHei'] # Use SimHei font

plt.rcParams['axes.unicode\_minus'] = False # Solve the issue with negative sign display

# Detect the file encoding

file\_path = r" "

with open(file\_path, 'rb') as f:

result = chardet.detect(f.read())

print(result)

# Load the data with the detected encoding

encoding = result['encoding']

data = pd.read\_csv(file\_path, encoding=encoding)

# View the structure of the data

print(data.head())

# 2. Data preprocessing

# Assume your dataset contains columns such as "concentration", "temperature", "dose", and "responsivity"

# Select features and target variable

X = data[['concentration(ppm)', 'temperature', 'Dose(mol%)']] # Feature columns

y = data['response'] # Target column (responsivity)

# 3. Split the dataset

# Split the dataset into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 4. Create and train a Gradient Boosting Regressor model

model = GradientBoostingRegressor(n\_estimators=100, learning\_rate=0.1, max\_depth=3)

model.fit(X\_train, y\_train)

# 6. Create test data with values for prediction

test\_data = pd.DataFrame([(10, 25, 0), (20, 25, 0), (30, 25, 0), (40, 25, 0), (50, 25, 0),

(10, 25, 1), (20, 25, 1), (30, 25, 1), (40, 25, 1), (50, 25, 1),

(10, 25, 3), (20, 25, 3), (30, 25, 3), (40, 25, 3), (50, 25, 3),

(10, 25, 5), (20, 25, 5), (30, 25, 5), (40, 25, 5), (50, 25, 5)],

columns=['concentration(ppm)', 'temperature', 'Dose(mol%)'])

# 7. Predict the responsivity

predictions = model.predict(test\_data)

# 8. Add the predicted results to the test data

test\_data['predicted\_responsivity'] = predictions

# 9. Output the prediction results

print(test\_data)

# 10. Visualization: Plot predicted responsivity at different dopant levels

plt.figure(figsize=(10, 6))

# Plot predicted results at different dopant levels

for dopant in [0, 1, 3, 5]:

subset = test\_data[test\_data['Dose(mol%)'] == dopant]

plt.plot(subset['concentration(ppm)'], subset['predicted\_responsivity'], label=f'Dose {dopant}%', marker='o')

# Add title and labels

plt.title("Concentration vs Predicted Responsivity at Different Dopant Levels", fontsize=14)

plt.xlabel("Concentration (ppm)", fontsize=12)

plt.ylabel("Predicted Responsivity", fontsize=12)

# Add legend

plt.legend()

# Show the plot

plt.show()