LAB Logbook

Lab 1

**CODE:**

import numpy as np

SID = 23750107

last\_two\_digits = int(str(SID)[-2:])

if last\_two\_digits < 10:

    num\_elements = last\_two\_digits + 100

else:

    num\_elements = last\_two\_digits

a = np.arange(num\_elements)

a\_2d = a.reshape(1, -1)

print(a\_2d)

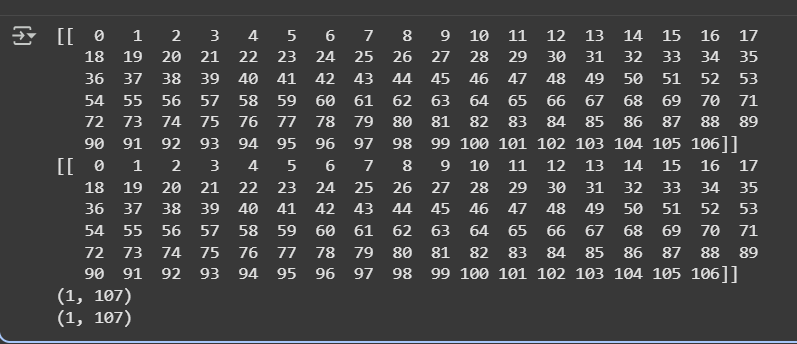
b = a\_2d.copy()

print(b)

print(a\_2d.shape)

print(b.shape)

**OUTPUT:**



Lab 2

**CODE:**

import pandas as pd

data = pd.DataFrame({

    "relationship": ["Husband", "Wife", "Own-child", "Not-in-family", "Husband", "Wife"],

    "hours-per-week": [40, 50, 60, 20, 55, 45]

})

n = 10

def reduce\_hours(x):

    return x - n

data["reduced\_hours"] = data["hours-per-week"].apply(reduce\_hours)

grouped\_original=data.groupby(["relationship","hours-per-week"]).size().reset\_index(name="count")

grouped\_reduced=data.groupby(["relationship", "reduced\_hours"]).size().reset\_index(name="count")

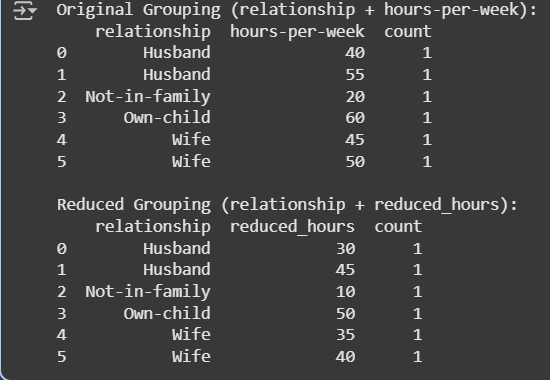
print("Original Grouping (relationship + hours-per-week):")

print(grouped\_original, "\n")

print("Reduced Grouping (relationship + reduced\_hours):")

print(grouped\_reduced)

**OUTPUT:**

****

Lab 3

**CODE:**

import pandas as pd

import matplotlib.pyplot as plt

# Example DataFrame with columns 0-9

# Replace this with your actual dataset

data = pd.DataFrame({

0: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

1: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1],

2: [5, 4, 3, 2, 1, 0, -1, -2, -3, -4],

3: [2, 3, 2, 3, 2, 3, 2, 3, 2, 3],

4: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19],

5: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20],

6: [3, 6, 9, 12, 15, 18, 21, 24, 27, 30],

7: [1, 4, 7, 10, 13, 16, 19, 22, 25, 28],

8: [0, 1, 0, 1, 0, 1, 0, 1, 0, 1],

9: [5, 3, 1, -1, -3, -5, -7, -9, -11, -13]

})

# Select the two columns from your SID

col1, col2 = 0, 1

plt.figure(figsize=(8,6))

plt.scatter(data[col1], data[col2], c='blue', label=f'Col {col1} vs Col {col2}')

plt.xlabel(f'Column {col1}')

plt.ylabel(f'Column {col2}')

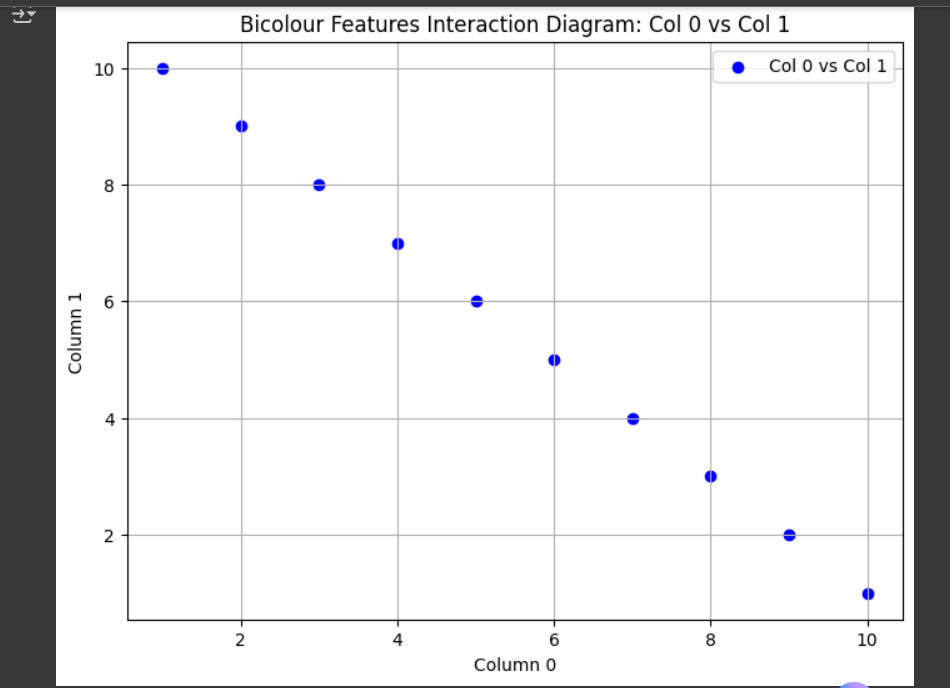
plt.title(f'Bicolour Features Interaction Diagram: Col {col1} vs Col {col2}')

plt.legend()

plt.grid(True)

plt.show()

**OUTPUT:**



Lab 4

**CODE:**

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import Adam

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

from sklearn.metrics import mean\_absolute\_error

from sklearn.preprocessing import MinMaxScaler

X = np.random.rand(1000, 10)  # 10 input features

y = np.random.rand(1000, 1)   # 1 target variable

scaler = MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

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

model = Sequential()

model.add(Dense(10, activation='relu', input\_shape=(X.shape[1],)))  # First hidden layer

model.add(Dense(5, activation='relu'))  # Second hidden layer

model.add(Dense(1))  # Output layer

model.compile(optimizer=Adam(), loss='mae', metrics=['mae'])

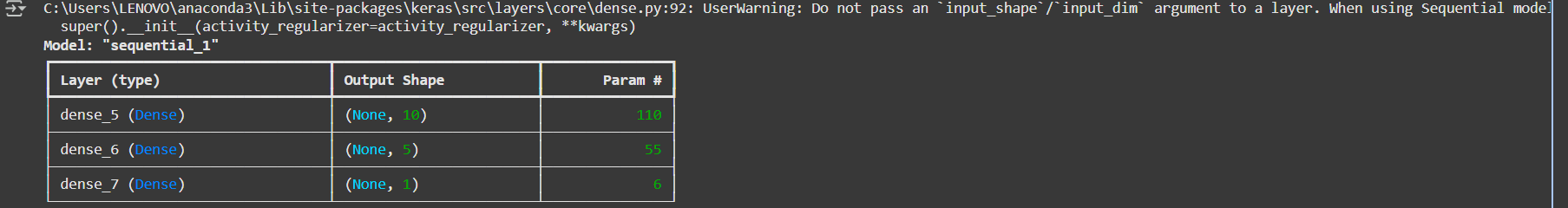
model.summary()

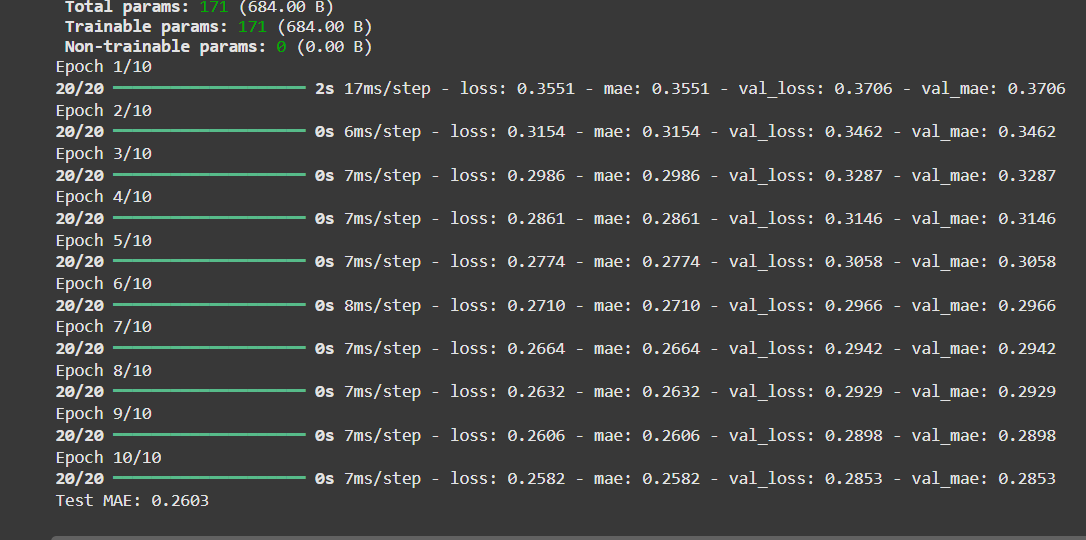
history = model.fit(X\_train, y\_train, epochs=10, validation\_split=0.2)

mae = model.evaluate(X\_test, y\_test, verbose=0)[1]

print(f"Test MAE: {mae:.4f}")

**OUTPUT:**

****

****

Lab 5

**CODE:**

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.datasets import cifar10

import matplotlib.pyplot as plt

# Load dataset

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0

# Build CNN

model = models.Sequential([

    layers.Conv2D(32, (5, 5), activation='relu', input\_shape=(32, 32, 3)),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (5, 5), activation='relu'),

    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),

    layers.Dense(64, activation='relu'),

    layers.Dense(10)

])

# Compile model

model.compile(optimizer='adam',

              loss='mean\_squared\_error',

              metrics=['mae'])

# Model summary

model.summary()

# Train CNN

history = model.fit(

    x\_train, y\_train,

    epochs=10,

    batch\_size=50,

    validation\_data=(x\_test, y\_test)

)

# Evaluate

test\_loss, test\_mae = model.evaluate(x\_test, y\_test, verbose=2)

print(f"\nTest MAE: {test\_mae:.4f}")

# Plot MAE

plt.figure(figsize=(8, 5))

plt.plot(history.history['mae'], label='Train MAE')

plt.plot(history.history['val\_mae'], label='Validation MAE')

plt.title('Model MAE During Training')

plt.xlabel('Epoch')

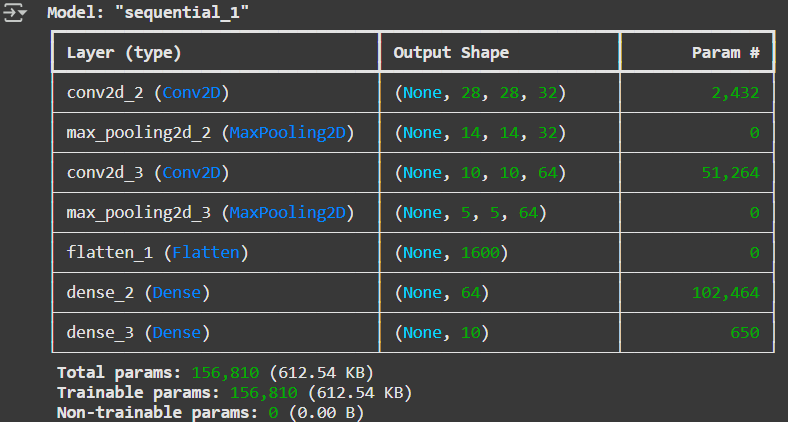
plt.ylabel('MAE')

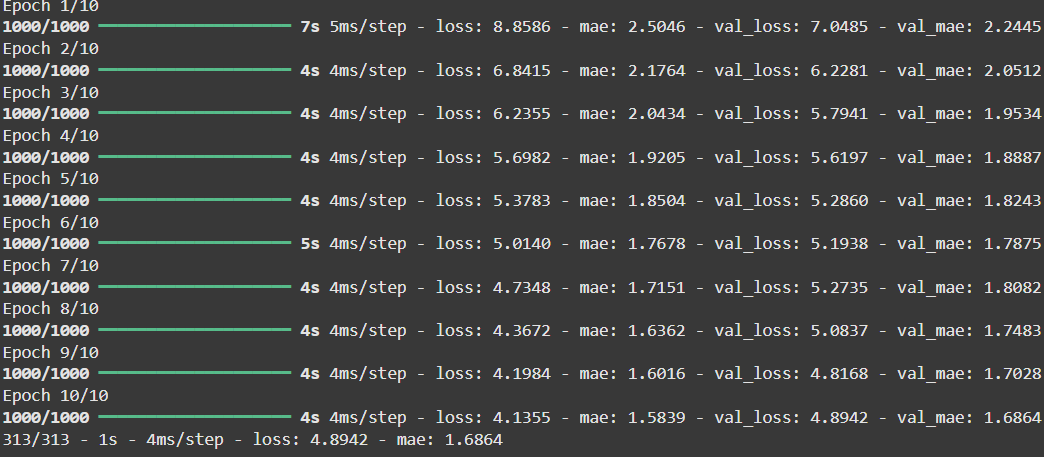
plt.legend()

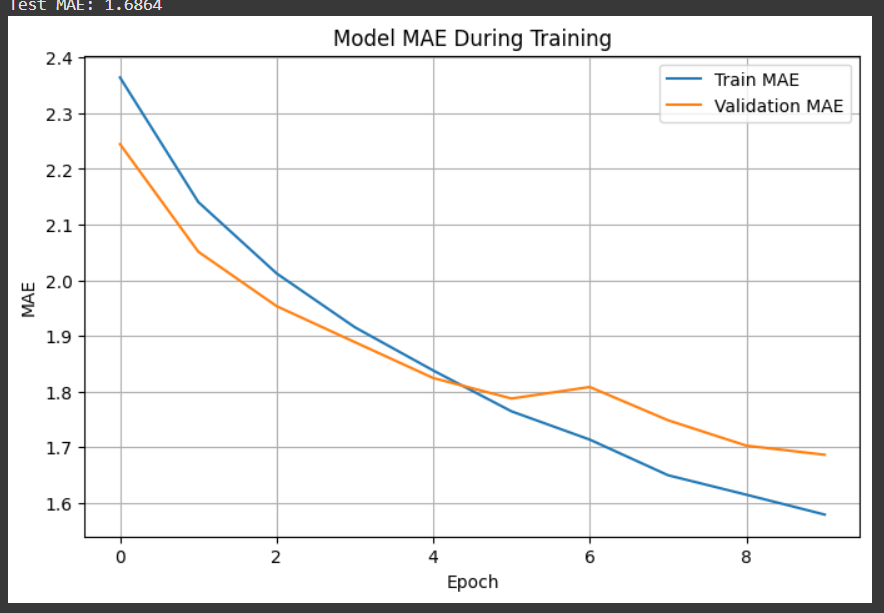
plt.grid(True)

plt.show()

**OUTPUT:**

****

****

****

Lab 6

Lab 7

Lab 8

Lab 9

Lab 10

Lab 11

Lab 12