تمرین دوم درس یادگیری ژرف(لی نت 5 پیش بینی)

رضا احمدی

1.واردکردن کتابخانه ها

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error ,r2\_score

from keras.models import Sequential

from keras.layers import Conv1D ,MaxPool1D, Flatten, Dense

from keras.layers import Dropout, BatchNormalization

from keras.optimizers import Nadam

from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

from keras.regularizers import l1\_l2

2.خواندن داده ها و چایگزینی داده های گم شده

data = pd.read\_excel('mlp1.xlsx')

data.fillna(method='ffill', inplace=True)

3.حذف داده های پرت

Q1 = data.quantile(0.25)

Q3 = data.quantile(0.75)

IQR = Q3 - Q1

data = data[~((data < (Q1 - 1.5 \* IQR)) | (data > (Q3 + 1.5 \* IQR))).any(axis=1)]

چارک اول (Q1)وسوم (Q3) حساب کرده

بازه ی قابل قبول (IQR) مشخص کردی

داده هایی که خیلی دور از q1 و Q3 بودن (Outlier) رو حذف کرده

4.جدا کردن ویژگی ها و هدف

x = data.iloc[:, :-1].values

y = data.iloc[:, -1].values

5.نرمال سازی داده ها

scaler = StandardScaler()

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

x\_scaled = x\_scaled.reshape((x\_scaled.shape[0],x\_scaled.shape[1],1))

استاندارد سازی داده ها (میانگین 0 و واریانس 1)

6.تقسیم داده ها

x\_train, x\_temp, y\_train, y\_temp = train\_test\_split(x\_scaled, y, test\_size=0.3, random\_state=42)

x\_val, x\_test, y\_val, y\_test = train\_test\_split(x\_temp, y\_temp, test\_size=0.5, random\_state=42)

70 درصد آموزش و 30 درصد موقت

از موقت نصف برای اعتبار سنجی و نصف دیگر برای تست نهایی

7.تعریف مدل

model = Sequential([

    Conv1D(filters=6, kernel\_size=5, strides=1, activation='tanh', padding='same', input\_shape=(x\_train.shape[1], 1), kernel\_regularizer=l1\_l2(l1=0.001, l2=0.01)),

    BatchNormalization(),

    MaxPool1D(pool\_size=2, strides=2),

    Dropout(0.2),

    Conv1D(filters=16, kernel\_size=3, strides=1, activation='tanh', padding='valid', kernel\_regularizer=l1\_l2(l1=0.001, l2=0.01)),

    BatchNormalization(),

    MaxPool1D(pool\_size=2, strides=2),

    Dropout(0.2),

    Flatten(),

    Dense(units=120, activation='tanh', kernel\_regularizer=l1\_l2(l1=0.001, l2=0.01)),

    Dropout(0.2),

    BatchNormalization(),

    Dense(units=84, activation='tanh',  kernel\_regularizer=l1\_l2(l1=0.001, l2=0.01)),

    Dropout(0.2),

    BatchNormalization(),

    Dense(units=1)

])

لی نت 5 ابتدا از یک لایه پیچشی با 6 فیلتر و بعد لایه پولینگ دوباره لایه پیچشی با 16 فیلتر و دوباره لایه پولینگ سپس مسطح سازی انجام میشود و سه لایه تماما متصل به ترتیب با 120 و 84 و 1 نورون

L1 : در این روش، به تابع هزینه یک جریمه اضافه میکنیم که برابر با مجموع قدر مطلق وزن ها است

هدف:تا حد ممکن وزن های غیر ضروری را صفر کند و مدلی با ویژگی های مهم تر و ساده تر

L2: در این روش، به تابع هزینه یک جریمه اضافه می شود که برابر با مجموع مربع وزن ها است

هدف: جلوگیری از بزرگ شدن بیش از خد وزن ها و ساختن مدلی پایدارتر و مقاوم تر

Dropout: در این روش هر بار آموزش به صورت تصادفی تعدادی از نورون ها غیرفعال می شود

این کار باغث می شود مدل نتواند به ترتیب خاصی از نورون ها وابسنه شود و یادیری مقاوم (Robust) و عمومی تر شود.برای جلوگیری از اورفیت شدن شبکه

Batch Normalization: در طی این آموزش میانگین داده های هر Batch محاسبه شده و خروجی لایه ها بر اساس اسناندارد می شود

سرعت آموزش زیاد میشود و شبکه پایدار میشود

8. کامپایل مدل

optimizer = Nadam(learning\_rate=0.001)

model.compile(optimizer=optimizer, loss='mean\_squared\_error', metrics=['mae'])

بهینه سازی با Nadam (ترکیبی از Adam و Nestrov Momentum)

9.آموزش مدل

early\_stopping = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)

model\_checkpoint = ModelCheckpoint('best\_model.h5', monitor='val\_loss', save\_best\_only=True, mode='min')

reduc\_lc = ReduceLROnPlateau(monitor='val\_loss', factor=0.5, patience=3)

history = model.fit(x\_train, y\_train,

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

                    epochs=150,

                    batch\_size=32,

                    callbacks=[early\_stopping,model\_checkpoint,reduc\_lc],

                    verbose=1)

Early stopping: اگر مدل پیشرفتی نداشت ، آموزش رو زودتر متوقف میکند

ModelCheckpoint: بهترین مدل را ذخیره میکند

ReduceLROnPlateau:اگر مدل گیر کرد نرخ یادگیری را نصق میکند

10.پیش بینی

y\_pred = model.predict(x\_test).flatten()

errors = y\_pred - y\_test

خروجی مدل را روی داده های تست پیش بینی میکند

خطای هر پیش بینی محاسبه شده

11 ترسیم منحنی Loss

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

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

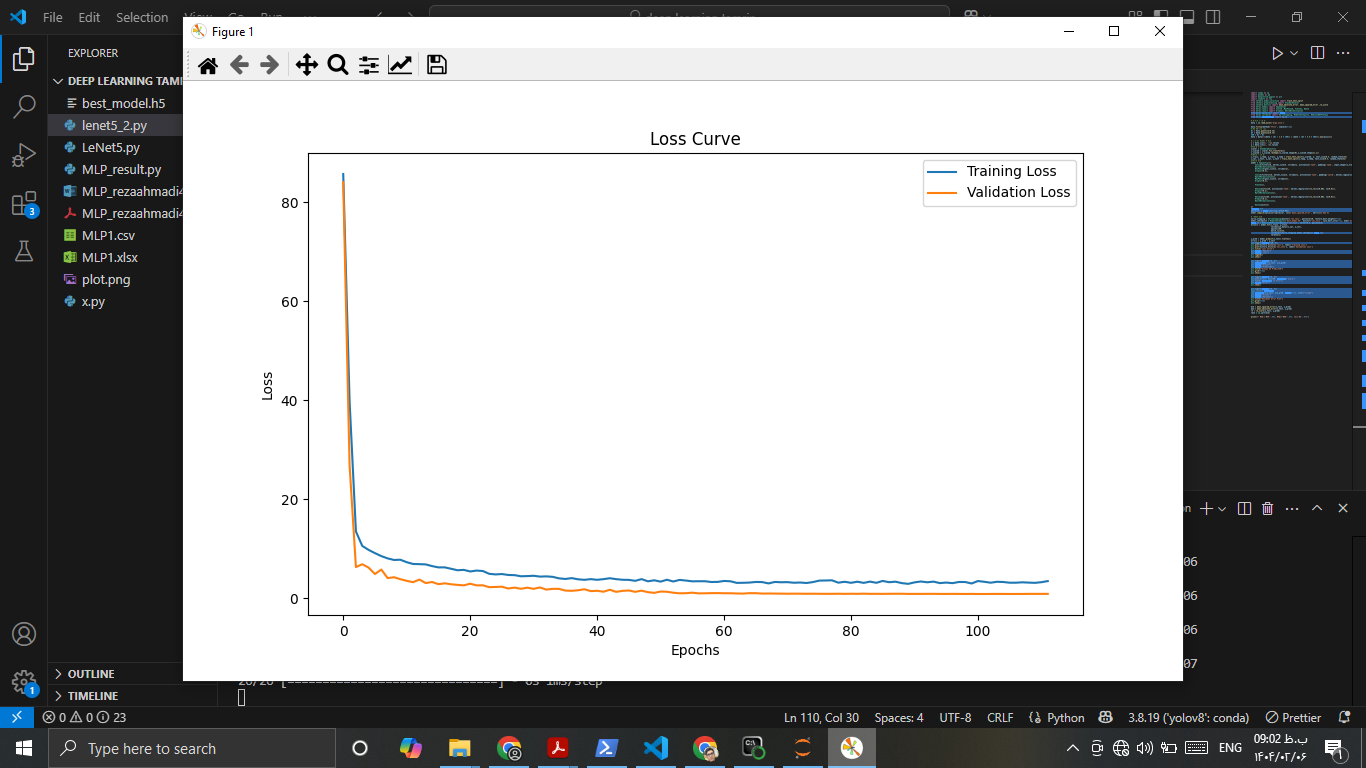
plt.title('Loss Curve')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()



روند کاهش خطای آموزش و اعتبار سنجی در طول دوره آموزش را رسم می کند

12.نمودار Actual VS Predicted

plt.figure(figsize=(6, 4))

sns.scatterplot(x=y\_test, y=y\_pred)

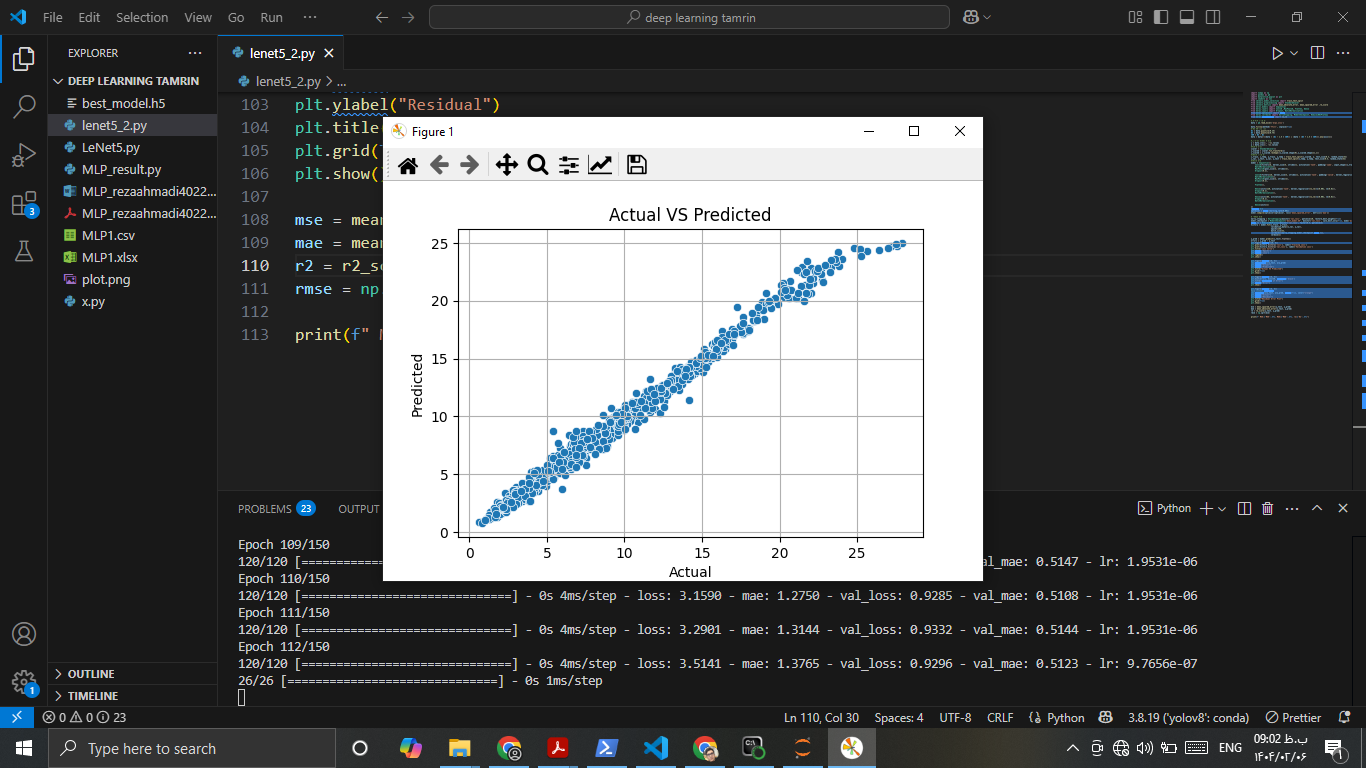
plt.xlabel("Actual")

plt.ylabel("Predicted")

plt.title("Actual VS Predicted")

plt.grid(True)

plt.show()



بررسی نطابق پیش بینی ها با داده واقغی

هرچه نزدیکتر به خط X=Y باشد بهتر است

13. هیستوگرام خطا

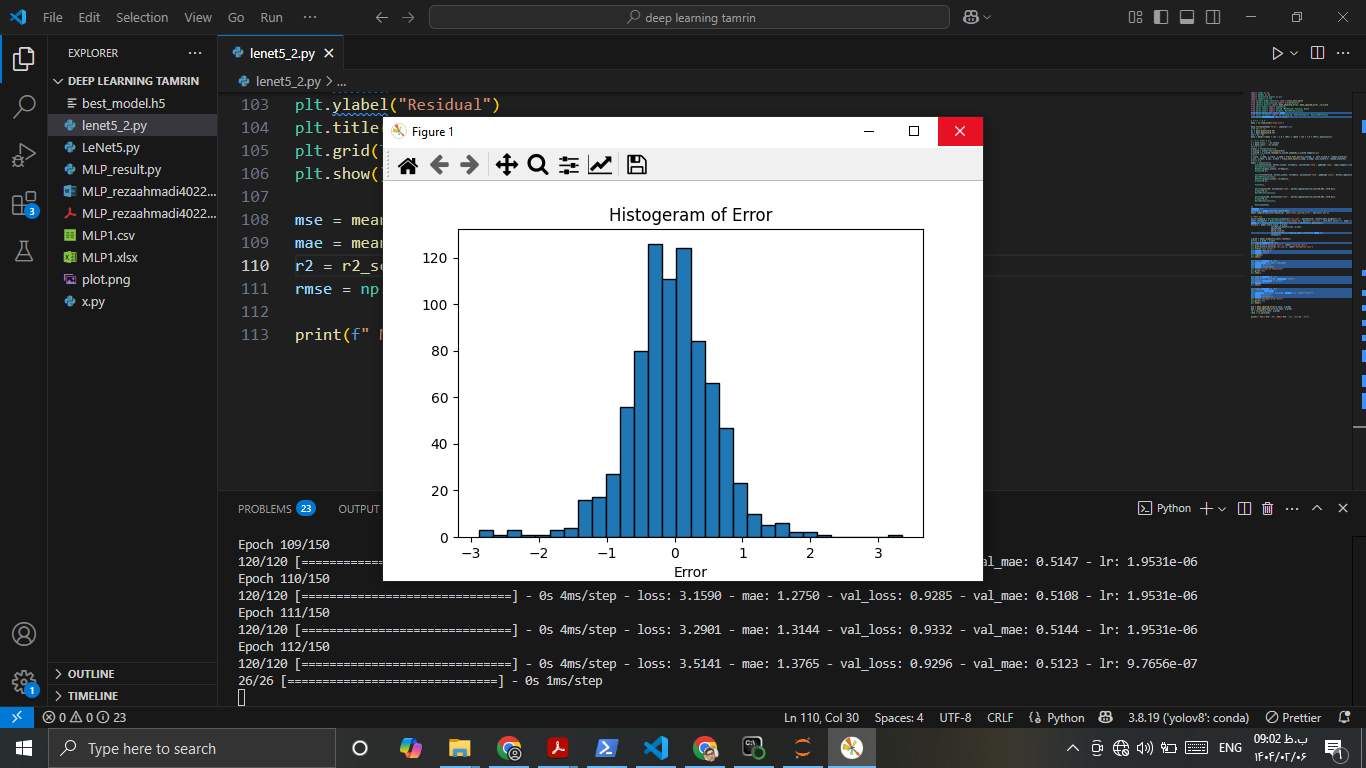
plt.figure(figsize=(6, 4))

plt.hist(errors, bins=30, edgecolor='black')

plt.title("Histogeram of Error")

plt.xlabel("Error")

plt.show()

توزیع خطاها را رسم کرده

اگر خطا حول صفر باشد مدل خوب یادگرفته

14.نمودار Residual Error

plt.figure(figsize=(6, 4))

#pip install statsmodel

sns.residplot(x=y\_test, y=y\_pred, lowess=True, color="orange")

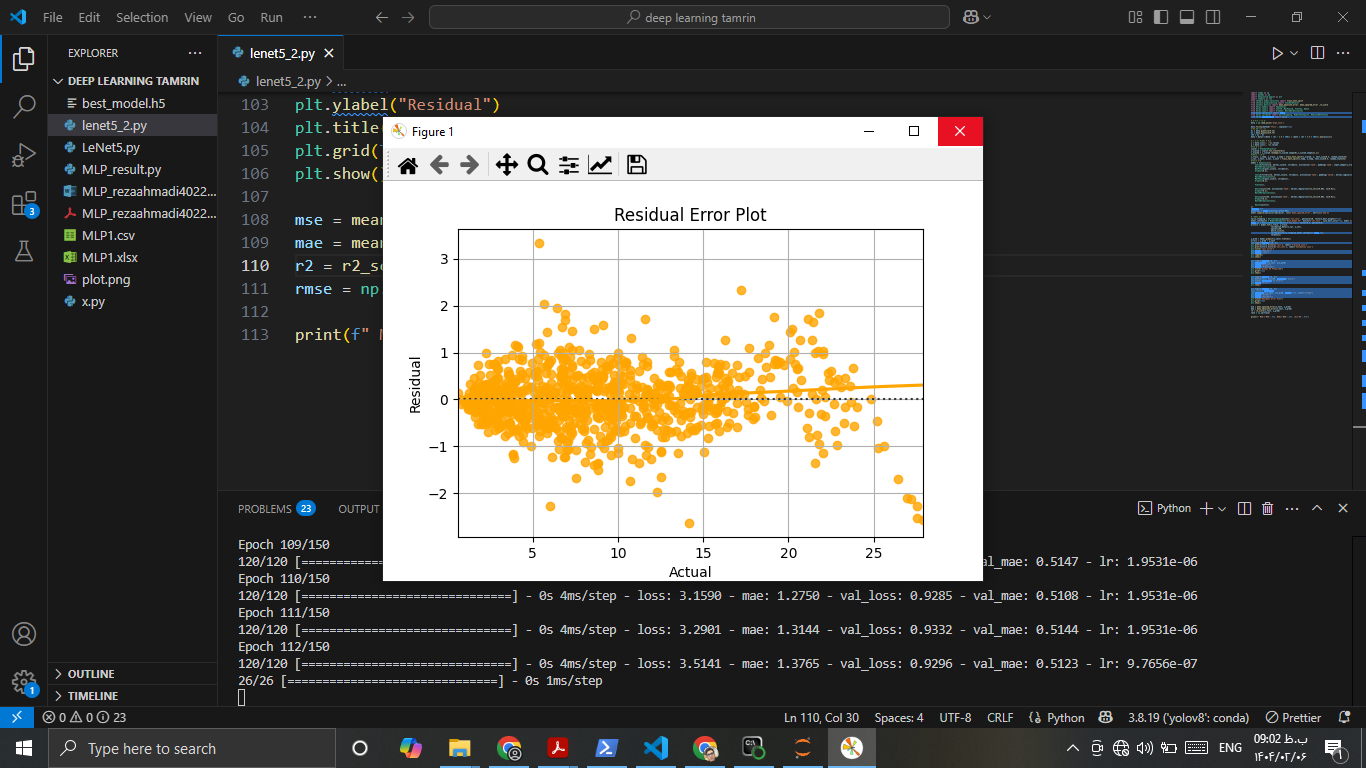
plt.xlabel("Actual")

plt.ylabel("Residual")

plt.title("Residual Error Plot")

plt.grid(True)

plt.show()



خطای باقی مانده را در مقابل مقادیر واقعی رسم کرده

باید حول محور صفر پخش شده باشد

15.محاسبه معیار های ارزیابی مدل

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

rmse = np.sqrt(mse)

print(f" MSE:{mse:.4f}, MAE:{mae:.4f}, R2\_score:{r2:.4f}, RMSE:{rmse:.4f}")

MSE:0.3880, MAE:0.4620, R2\_score:0.9888, RMSE:0.6229