تمرین دوم درس یادگیری ژرف(لی نت ۵ پیش بینی)

رضا احمدی

۱.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
```

۲.خواندن داده ها و چایگزینی داده های گم شده

```
data = pd.read_excel('mlp1.xlsx')
data.fillna(method='ffill', inplace=True)
```

۳.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

۴.جدا کردن ویژگی ها و هدف

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

۵.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 ۰ و واریانس ۱)

۶.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)
```

۷۰ درصد آموزش و ۳۰ درصد موقت

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

۷.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 ۵ ابتدا از یک لایه پیچشی با ۶ فیلتر و بعد لایه پولینگ دوباره لایه پیچشی با ۱۶ فیلتر و دوباره لایه پولینگ سپس مسطح سازی انجام میشود و سه لایه تماما متصل به ترتیب با ۱۲۰ و ۸۴ و ۱ 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 وابسته 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 استاندارد می شود

سرعت آموزش زیاد میشود و شبکه پایدار میشود

۸. 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)

۹.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

۱۰.پیش بینی

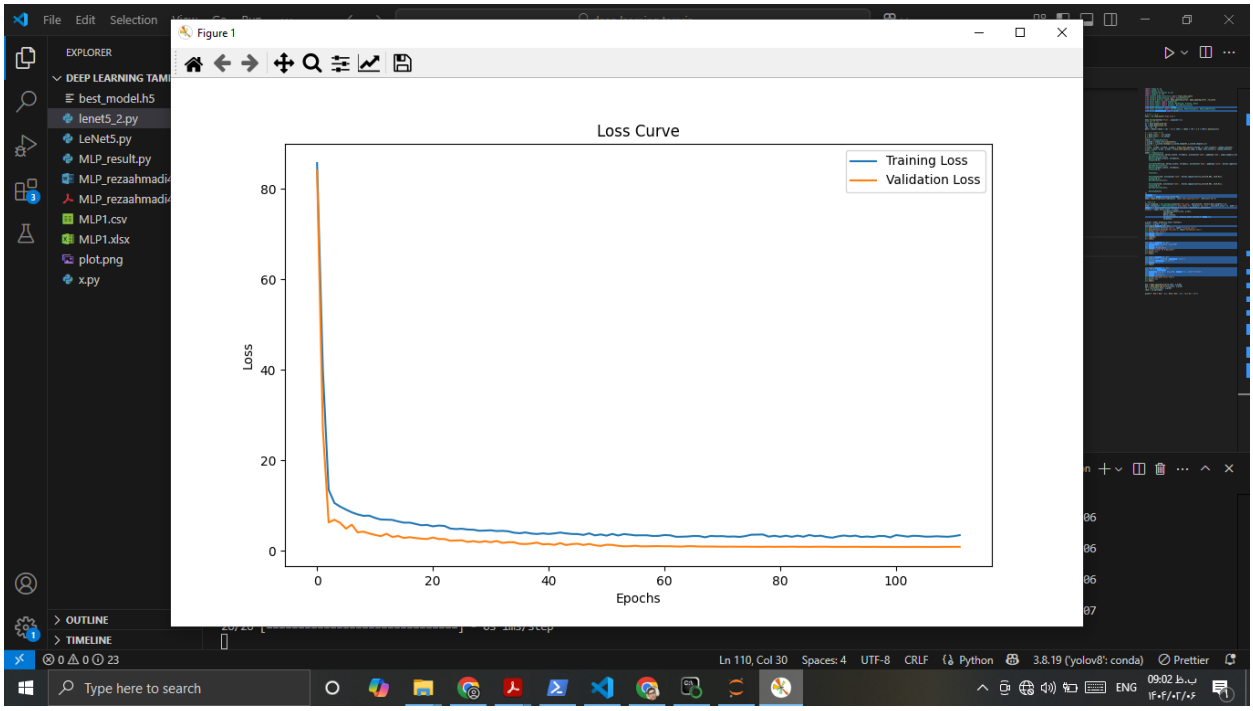
```
y_pred = model.predict(x_test).flatten()
errors = y_pred - y_test
```

خروجی مدل را روی داده های تست پیش بینی میکند

خطای هر پیش بینی محاسبه شده

۱۱ ترسیم منحنی Loss

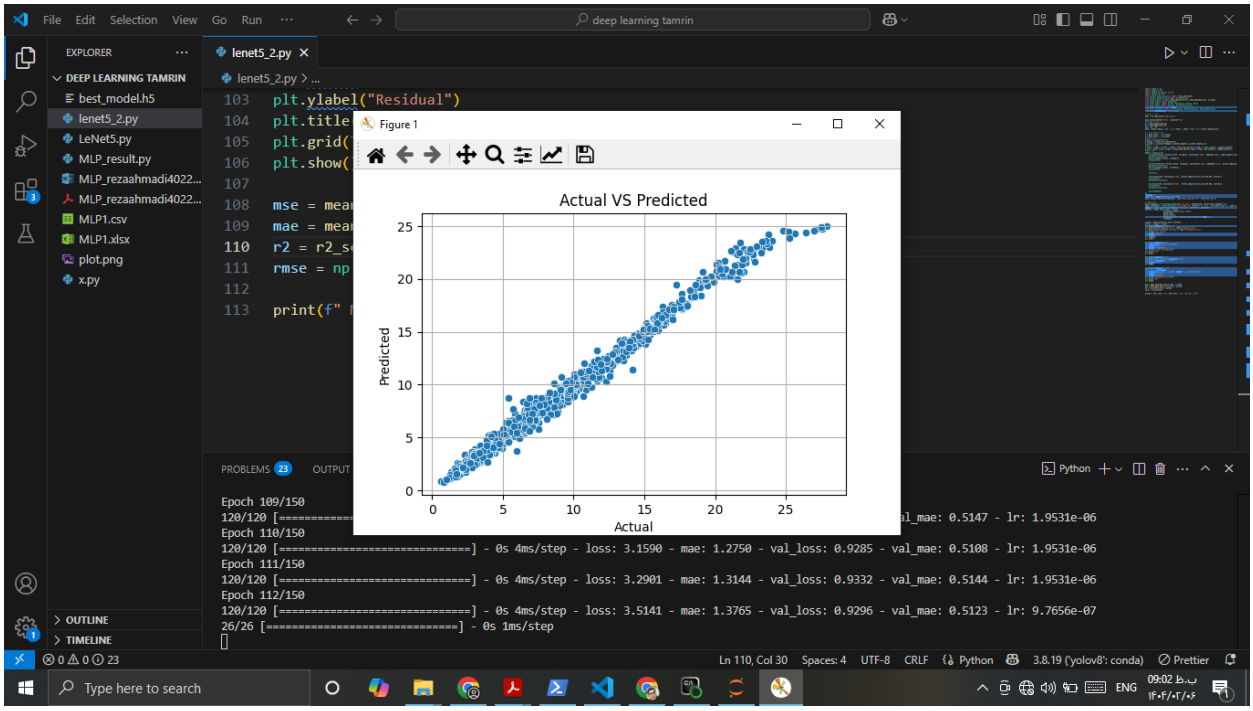
```
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Curve')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



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

۱۲.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()
```

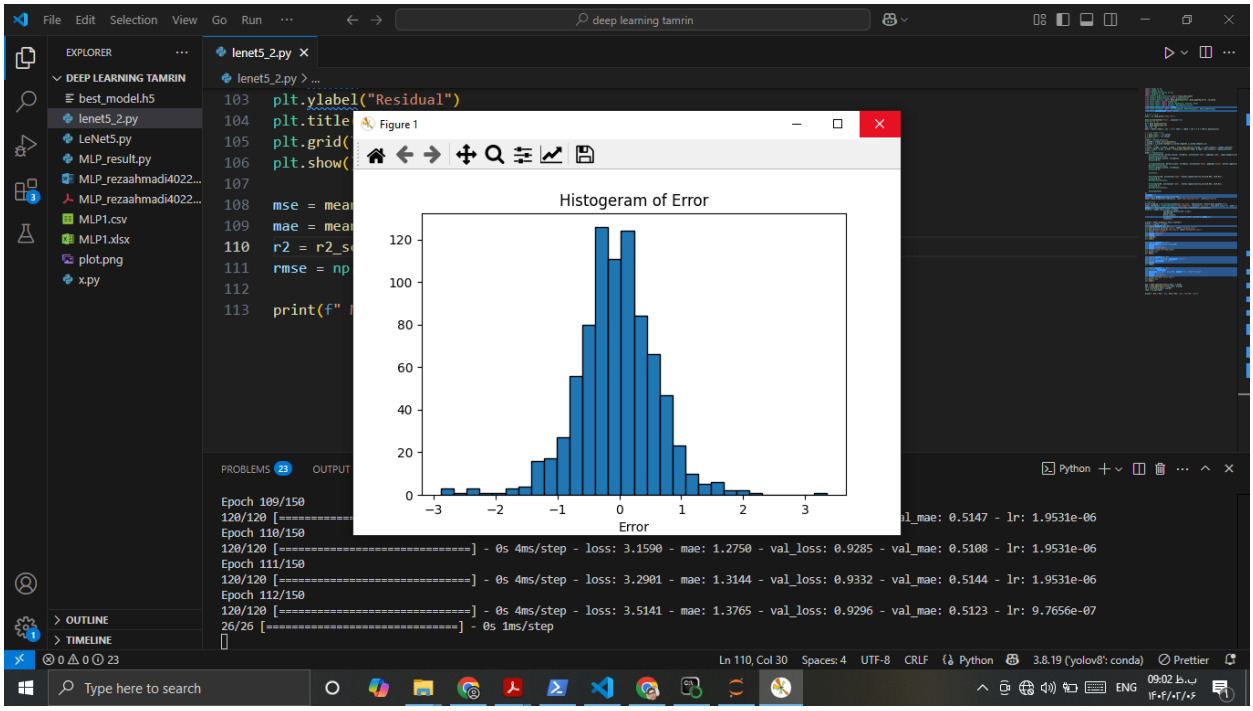


بررسی نطابق پیش بینی ها با داده واقغی

هرچه نزدیکتر به خط X=Y باشد بهتر است

۱۳. هیستوگرام خطا

```
plt.figure(figsize=(6, 4))
plt.hist(errors, bins=30, edgecolor='black')
plt.title("Histogeram of Error")
plt.xlabel("Error")
plt.show()
```

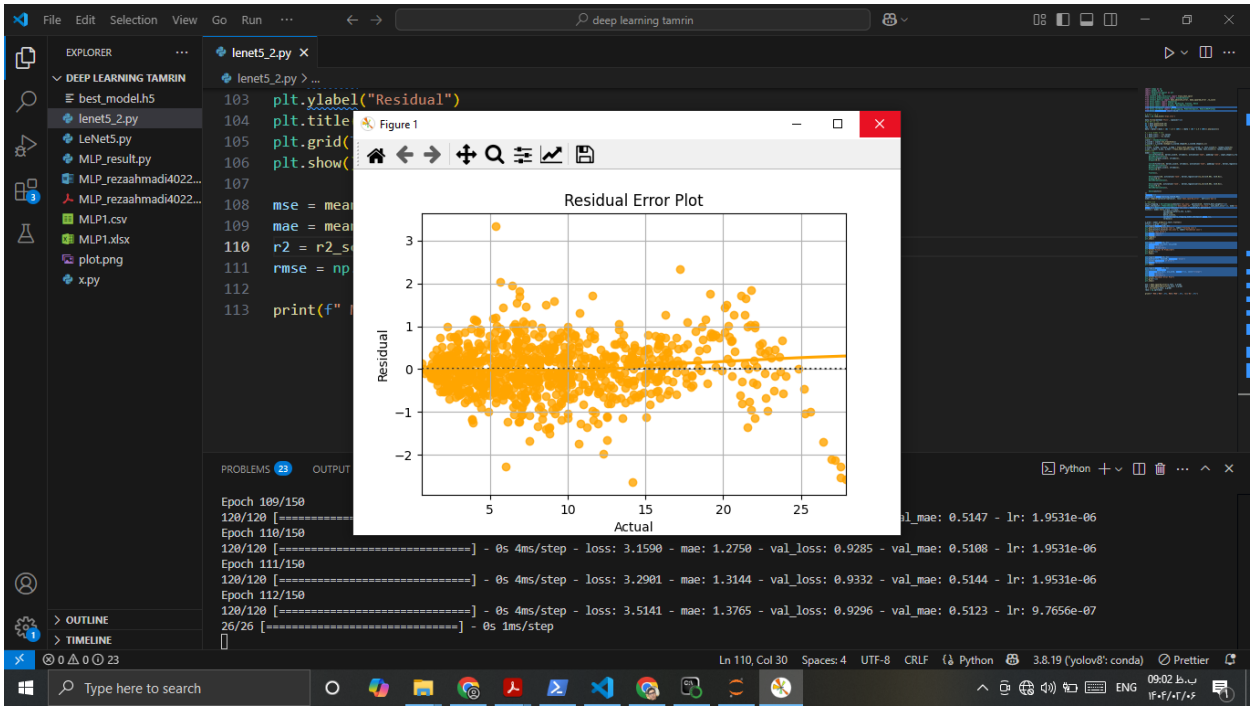


توزیع خطاها را رسم کرده

اگر خطا حول صفر باشد مدل خوب یادگرفته

۱۴.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

۱۵.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