```
import os, time, math
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from IPython.display import clear_output, display
```

Double-click (or enter) to edit

```
p="salary_data.csv"
if not os.path.exists(p): raise FileNotFoundError("Upload salary_data.csv")
df=pd.read_csv(p)
```

```
yrs=[c for c in df.columns if 'year' in c.lower() or 'experience' in c.lower()]
sal=[c for c in df.columns if 'salary' in c.lower() or 'sal' in c.lower()]
Xcol=yrs[0]; Ycol=sal[0]
X=df[Xcol].astype(float).values.reshape(-1,1); y=df[Ycol].astype(float).values.reshape(-1,1)
```

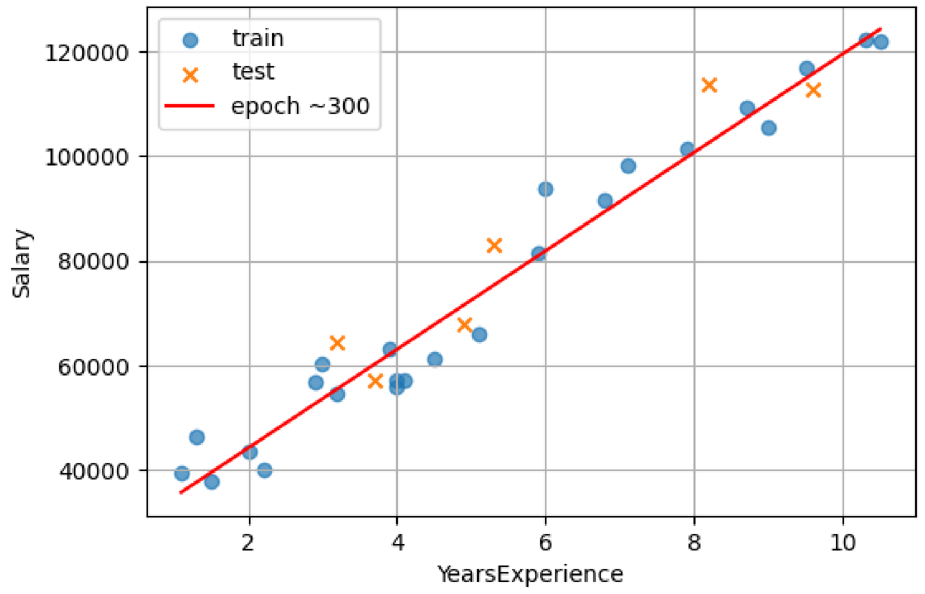
```
X_tr,X_te,y_tr,y_te = train_test_split(X,y,test_size=0.2,random_state=42)
Xm, Xs = X_tr.mean(), X_tr.std(ddof=0)
ym, ys = y_tr.mean(), y_tr.std(ddof=0)
Xn_tr=(X_tr-Xm)/Xs; Xn_te=(X_te-Xm)/Xs
yn_tr=(y_tr-ym)/ys

Xn_tr=Xn_tr.flatten(); Xn_te=Xn_te.flatten(); yn_tr=yn_tr.flatten()
```

```
def gd(X,y,lr,epochs,record=False):
    t0,t1=0.0,0.0; hist=[]
    m=len(y)
    for e in range(1,epochs+1):
        pred=t0+t1*X
        d0 = (1/m)*np.sum(pred-y)
        d1 = (1/m)*np.sum((pred-y)*X)
        t0 -= lr*d0; t1 -= lr*d1
        if record: hist.append((t0,t1))
    return t0,t1,hist
```

```
lrs=[0.001,0.005,0.01,0.03]; epochs_list=[50,100,300]
best=None; best_r2=-1
```

```
for lr in lrs:
    for ep in epochs_list:
        t0,t1,_=gd(Xn_tr,yn_tr,lr,ep,record=False)
        pred_n = t0 + t1*Xn_te
        pred = pred_n*ys + ym
        r2 = r2_score(y_te.flatten(), pred.flatten())
        if r2>best_r2:
            best_r2=r2; best={'lr':lr,'epochs':ep,'t0':t0,'t1':t1}
```

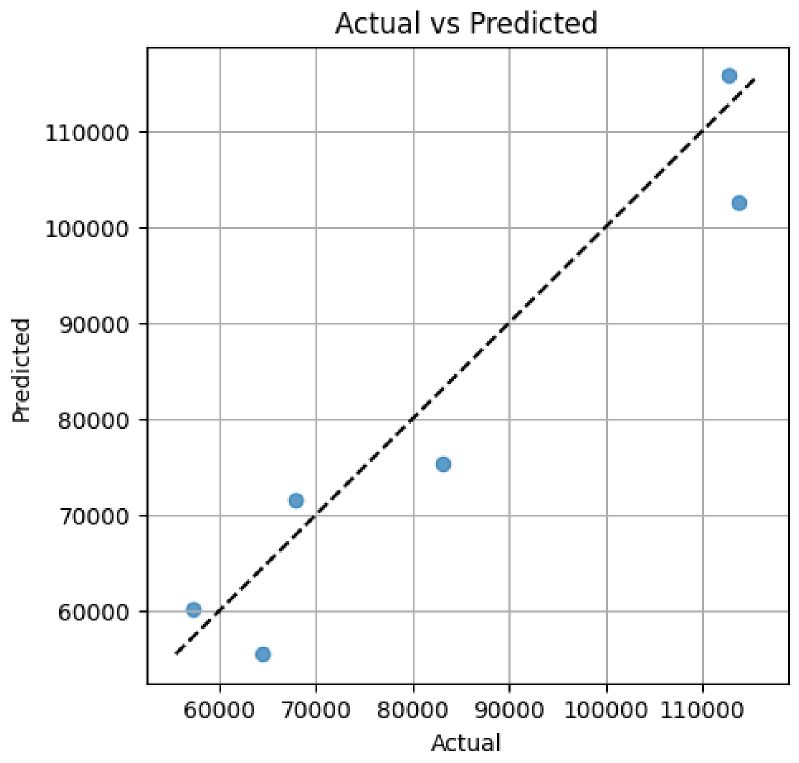
```
lr=best['lr']; epochs=best['epochs']
t0,t1,history = gd(Xn_tr,yn_tr,lr,epochs,record=True)
frames = history[::max(1,len(history)//60)] + [history[-1]]
xs = np.linspace(X.min(),X.max(),100)
for i,(th0,th1) in enumerate(frames,1):
    clear_output(wait=True)
    xs_n=(xs - Xm)/Xs
    ys_line = (th0 + th1*xs_n)*ys + ym
    plt.figure(figsize=(6,4))
    plt.scatter(X_tr,y_tr,label='train',alpha=0.7)
    plt.scatter(X_te,y_te,label='test',marker='x')
    plt.plot(xs,ys_line,color='red',label=f'epoch ~{int((i/len(frames))*epochs)}')
    plt.xlabel(Xcol); plt.ylabel(Ycol); plt.legend(); plt.grid(True)
    display(plt.gcf()); plt.close(); time.sleep(0.05)
```



```
pred_n_final = t0 + t1*Xn_te
pred_final = (pred_n_final*ys + ym).flatten()
y_true = y_te.flatten()
print(f"Best lr={lr}, epochs={epochs}")
print("R²:", r2_score(y_true,pred_final))
print("MAE:", mean_absolute_error(y_true,pred_final))
print("RMSE:", math.sqrt(mean_squared_error(y_true,pred_final)))
```

```
Best lr=0.03, epochs=300
R²: 0.9024406059472045
MAE: 6286.2005005056635
RMSE: 7059.245196238471
```

```
plt.figure(figsize=(5,5))
plt.scatter(y_true,pred_final,alpha=0.7); mn=min(y_true.min(),pred_final.min()); mx=max(y_true.max(),pred_final.max())
plt.plot([mn,mx],[mn,mx],'k--'); plt.xlabel("Actual"); plt.ylabel("Predicted"); plt.title("Actual vs Predicted"); plt.grid(True); plt.show()
```


Actual vs Predicted
Predicted
Actual