```python
import sys
import numpy as np
import pandas as pd
import openpyxl
import matplotlib.pyplot as plt
from pathlib import Path

# === Define Versions ===
print(sys.executable)
print("pandas:", pd.__version__, "openpyxl:", openpyxl.__version__)

# === Load dataset ===
candidate = Path(r"C:\Users\jaishva\Downloads\Teamwork-regress w 2 losses
(dataset)-1.xlsx")
df = pd.read_excel(candidate, engine="openpyxl")
print(df.head())

# === Preparing data (y = w*x + b) ===
X = np.c_[df["x"].to_numpy(), np.ones(len(df))]
y = df["y"].to_numpy().reshape(-1,1)


# === Gradient Descent Implementation ===
def gd_linear_regression(X, y, loss="mse", lr=0.01, delta=1.0, max_iters=1000,
                         tol_param=1e-6, tol_loss=1e-8):
    """
    Gradient descent for linear regression.
    Supports MSE and Huber losses.
    """
    n, d = X.shape
    theta = np.zeros((d,1))  # init params
    losses, param_deltas = [], []
    k_param = k_loss = None

    for k in range(max_iters):
        y_pred = X @ theta
        error  = y_pred - y

        # --- Loss + gradient ---
        if loss == "mse":
            L = np.mean(error**2)
            grad = (2/n) * (X.T @ error)
        elif loss == "huber":
            abs_err = np.abs(error)
            quadratic = abs_err <= delta
            L = np.mean(
                np.where(quadratic,
                         0.5*error**2,
                         delta*(abs_err - 0.5*delta))
```

```
            )
            grad = (X.T @ np.where(quadratic, error, delta*np.sign(error))) /
n
        else:
            raise ValueError("loss must be 'mse' or 'huber'")

        losses.append(L)

        # --- Updates ---
        theta_new = theta - lr*grad
        param_delta = np.linalg.norm(theta_new - theta)
        param_deltas.append(param_delta)
        theta = theta_new

        # --- Stopping criteria ---
        if k_param is None and param_delta < tol_param:
            k_param = k
        if k_loss is None and k > 0 and abs(losses[-2] - losses[-1]) <
tol_loss:
            k_loss = k
        if k_param is not None or k_loss is not None:
            break

    return theta, losses, param_deltas, k_param, k_loss


# === Let us run our experiments ===
lrs   = [0.01, 0.05, 0.2]
delta = 0.5

for loss in ["mse", "huber"]:
    print(f"\n=== {loss.upper()} ===")
    plt.figure()
    for lr in lrs:
        theta, losses, param_deltas, k_param, k_loss = gd_linear_regression(
            X, y, loss=loss, lr=lr, delta=delta, max_iters=3000,
            tol_param=1e-6, tol_loss=1e-8
        )
        print(f"lr={lr:<4}  theta=[{theta[0,0]:.4f}, {theta[1,0]:.4f}]  "
              f"final_loss={losses[-1]:.6f}  "
              f"stop(param)={k_param}  stop(loss)={k_loss}  iters={len(losses)
}")

        # Visualize all LRs on one figure
        it = np.arange(len(losses))
        plt.plot(it, losses, label=f"lr={lr}")
        if k_param is not None and k_param < len(losses):
            plt.scatter([k_param], [losses[k_param]], marker='o')
        if k_loss is not None and k_loss < len(losses):
```

```
            plt.scatter([k_loss], [losses[k_loss]], marker='x')

    plt.xlabel("Iteration")
    plt.ylabel("Training loss")
    plt.title(f"Learning curves — {loss.upper()} (o: param tol, x: loss tol)")
    plt.legend()
    plt.grid(True, alpha=0.3)
    plt.show()

# === Let us now compare the values with Least Squares solution ===
theta_ls, *_ = np.linalg.lstsq(X, y, rcond=None)
print("LS  theta:", theta_ls.ravel())

theta_mse, *_ = gd_linear_regression(X, y, loss="mse", lr=0.05,
max_iters=3000)
print("GD  theta (MSE, lr=0.05):", theta_mse.ravel())

print("||GD - LS||:", np.linalg.norm(theta_mse - theta_ls))
```

Terminal Output

C:\Users\jaishva\Assignment\venv\Scripts\python.exe

pandas: 2.3.3 openpyxl: 3.1.5

	x	y
0	0.417411	0.841049
1	0.222108	0.556829
2	0.119865	0.518283
3	0.337615	0.788053
4	0.942910	1.067603

=== MSE ===

lr=0.01 theta=[0.7266, 0.4607] final_loss=0.011105 stop(param)=None stop(loss)=None iters=3000

lr=0.05 theta=[0.7305, 0.4589] final_loss=0.011102 stop(param)=None stop(loss)=721 iters=722

lr=0.2   theta=[0.7320, 0.4581] final_loss=0.011102 stop(param)=None stop(loss)=204 iters=205

=== HUBER ===

lr=0.01 theta=[0.6768, 0.4844] final_loss=0.005686 stop(param)=None stop(loss)=None iters=3000

lr=0.05 theta=[0.7276, 0.4603] final_loss=0.005552 stop(param)=None stop(loss)=1259 iters=1260

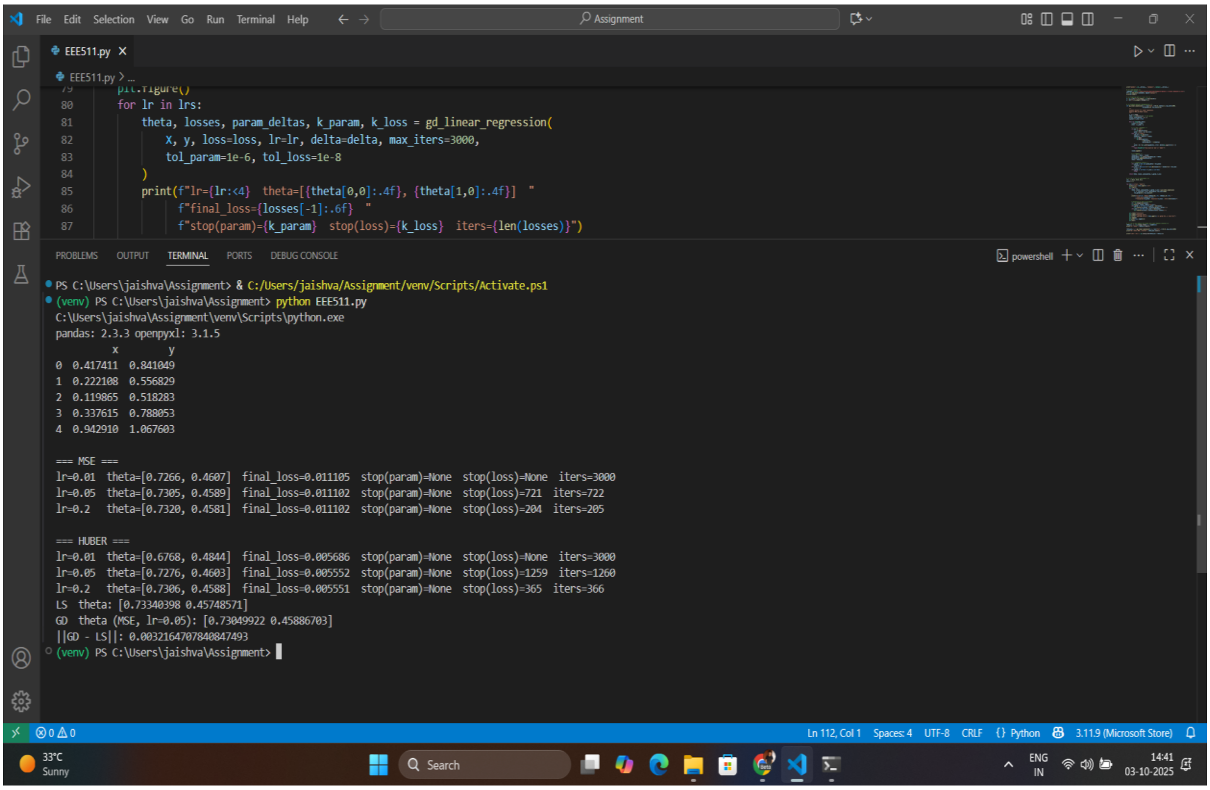
lr=0.2   theta=[0.7306, 0.4588] final_loss=0.005551 stop(param)=None stop(loss)=365 iters=366

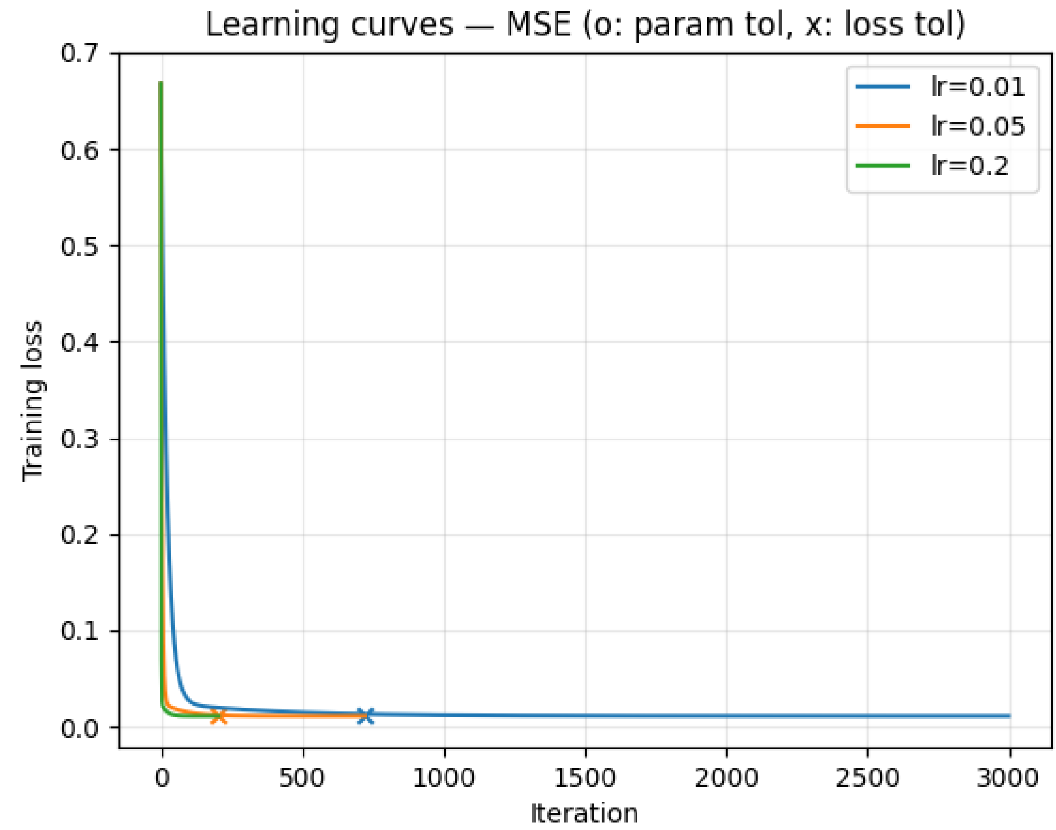
LS theta: [0.73340398 0.45748571]

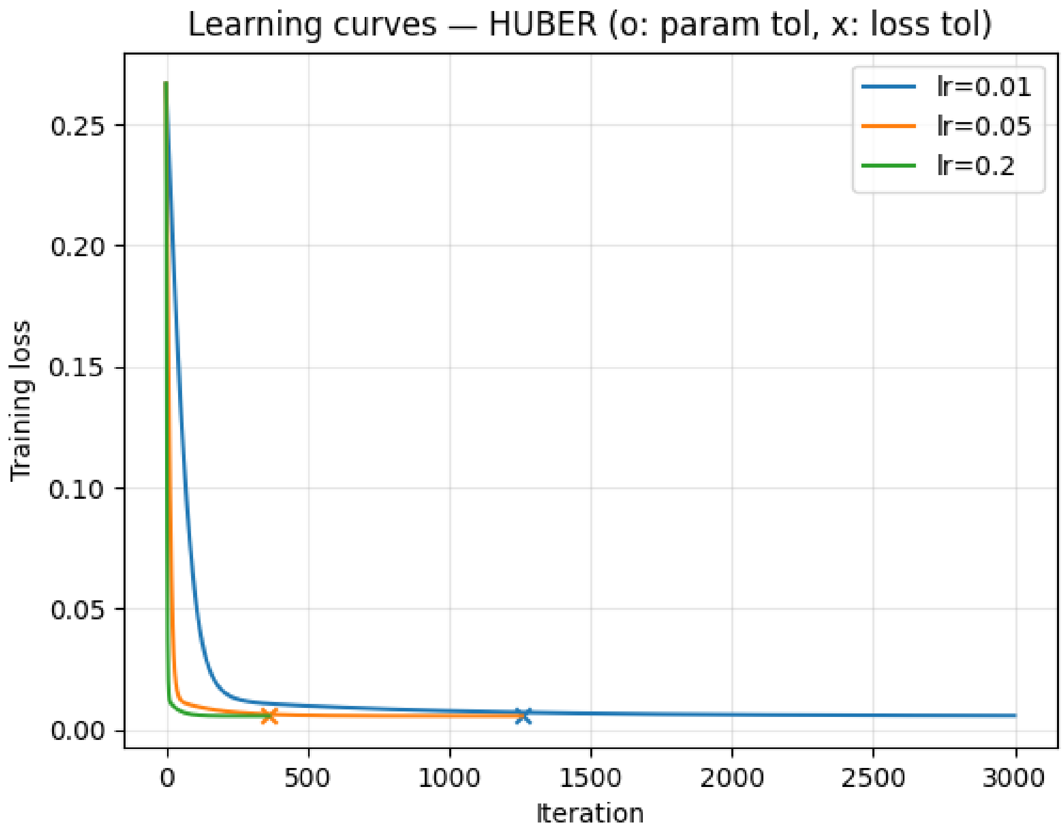
GD theta (MSE, lr=0.05): [0.73049922 0.45886703]

||GD - LS||: 0.0032164707840847493

Proof:-




Learning curves — MSE (o: param tol, x: loss tol)
Training loss
Iteration
lr=0.01
lr=0.05
lr=0.2
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
500
1000
1500
2000
2500
3000


Learning curves — HUBER (o: param tol, x: loss tol)
Training loss
Iteration
lr=0.01
lr=0.05
lr=0.2
0.25
0.20
0.15
0.10
0.05
0.00
0
500
1000
1500
2000
2500
3000