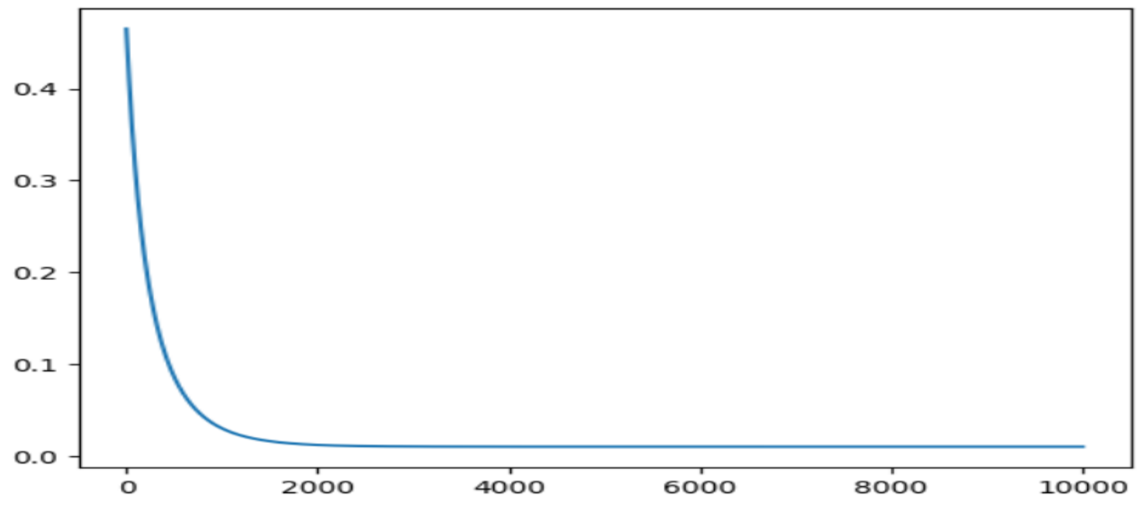
Source Code:

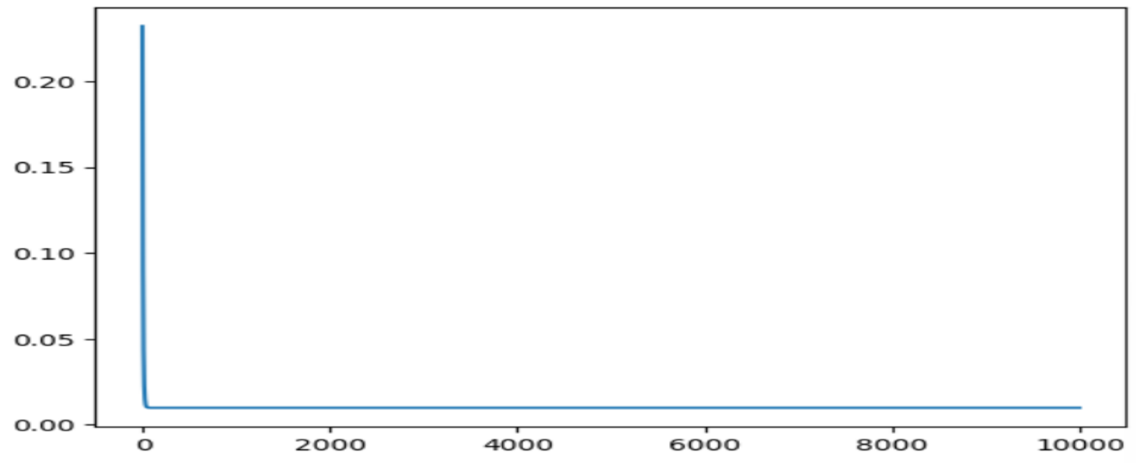
```
import pandas as pd
import numpy as np
import torch
import torch.optim as optim
import torch.nn as nn
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("/Housing.csv")
df.head()
df['mainroad'].dtype
X = df[['area', 'bedrooms', 'bathrooms', 'stories',  'parking']].values
y = df['price'].values
z = df[['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning','prefarea',
'furnishingstatus']]
X = df[['area', 'bedrooms', 'stories', 'parking']]
y = df['price'].values
z = df [['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'parking',
'prefarea', 'furnishingstatus']].apply(LabelEncoder().fit_transform)
X['mainroad']=z['mainroad']
X['guestroom']=z['guestroom']
X['basement']=z['basement']
X['hotwaterheating']=z['hotwaterheating']
X['airconditioning']=z['airconditioning']
X['parking']=z['parking']
X['prefarea']=z['prefarea']
X['furnishingstatus']=z['furnishingstatus']
X.head()
X.shape

X_mean = X.mean(axis=0)
X_std = X.std(axis=0)
X_norm = (X - X_mean) / X_std

X_norm.shape
```

```
y_min, y_max = y.min(), y.max()
y_norm = (y - y_min) / (y_max - y_min)
y_norm.shape
X_tensor = torch.tensor(X_norm.values, dtype=torch.float32)
y_tensor = torch.tensor(y_norm.reshape(-1, 1), dtype=torch.float32)

class SingleLayerModel(nn.Module):
    def __init__(self):
        super(SingleLayerModel, self).__init__()
        self.linear = nn.Linear(11, 1)  # 11 inputs -> 1 output
    def forward(self, x):
        z = self.linear(x)
        return z

model = SingleLayerModel()
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.0001)

epochs = 10000
loss_val=[]
for epoch in range(epochs):
    y_pred = model(X_tensor)
    loss = criterion(y_pred, y_tensor)
    print(f"Epoch: {epoch+1}, loss:{loss}")
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    loss_val.append(loss.item())
with torch.no_grad():
    final_preds_torch = model(X_tensor).numpy().flatten()
final_preds_denorm = final_preds_torch * (y_max - y_min) + y_min
plt.plot(loss_val)
plt.show()
for name, param in model.named_parameters():
    print(f"{name}: {param.data.numpy()}")
print("Final MSE Loss:", loss.item())
```

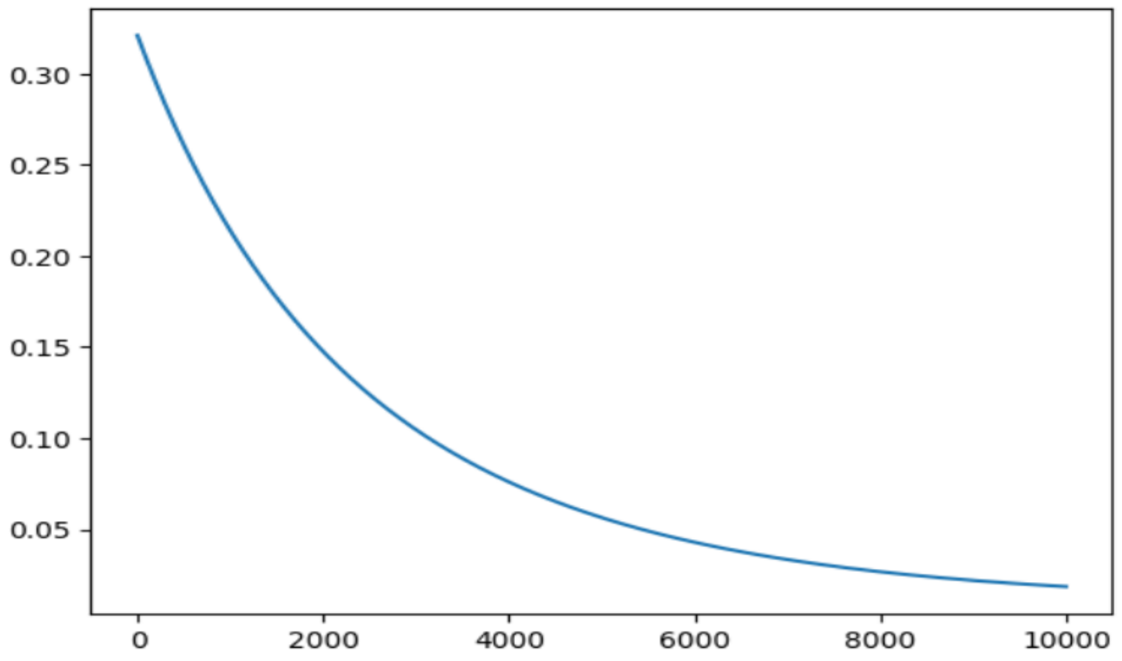
OUTPUT (lr =0.001):



OUTPUT (lr =0.04):



OUTPUT (lr =0.0001):



0.30
0.25
0.20
0.15
0.10
0.05
0
2000
4000
6000
8000
10000