Creating a Single-Layer Neural Network

```
import pandas as pd
import torch
import torch.nn as nn
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('/content/SomervilleHappinessSurvey2015.csv')
data.head()
```

	D	X1	X2	X3	X4	X5	X6
0	0	3	3	3	4	2	4
1	0	3	2	3	5	4	3
2	1	5	3	3	3	3	5
3	0	5	4	3	3	3	5
4	0	5	4	3	3	3	5

```
x = torch.tensor(data.iloc[:,1:].values).float()
y = torch.tensor(data.iloc[:,:1].values).float()
```

```
model = nn.Sequential(nn.Linear(6, 1),
                      nn.Sigmoid())
```

```
loss_function = torch.nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

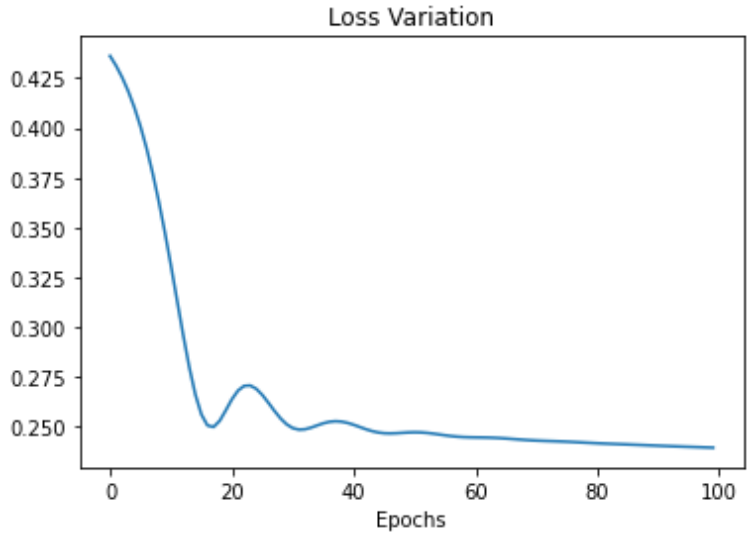
```
losses = []
for i in range(100):
    y_pred = model(x)
    loss = loss_function(y_pred, y)
    losses.append(loss.item())
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if i%10 == 0:
        print(loss.item())
```

```
0.43613865971565247
0.330939918756485
0.2638051509857178
0.24934566020965576
0.2510071098804474
```

```
0.24725835025310516
0.2446811944246292
0.24309661984443665
0.2417440563440323
0.24051088094711304
```

```
plt.plot(range(0,100), losses)
plt.title('Loss Variation')
plt.xlabel('Epochs')
plt.show()
```



Prediction of the release year of a song from audio features. Songs are mostly western, commercial tracks ranging from 1922 to 2011, with a peak in the year 2000s. See https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD

```
import pandas as pd
```

```
data = pd.read_csv("/content/YearPredictionMSD.txt", nrows=50000)
data.head()
```

	2001	49.94357	21.47114	73.07750	8.74861	-17.40628	-13.09905	-25.01202	-12.23
0	2001	48.73215	18.42930	70.32679	12.94636	-10.32437	-24.83777	8.76630	-0.92
1	2001	50.95714	31.85602	55.81851	13.41693	-6.57898	-18.54940	-3.27872	-2.35
2	2001	48.24750	-1.89837	36.29772	2.58776	0.97170	-26.21683	5.05097	-10.34
3	2001	50.97020	42.20998	67.09964	8.46791	-15.85279	-16.81409	-12.48207	-9.37
4	2001	50.54767	0.31568	92.35066	22.38696	-25.51870	-19.04928	20.67345	-5.19

5 rows × 91 columns

```
cols = data.columns

num_cols = data._get_numeric_data().columns

list(set(cols) - set(num_cols))
```

```
[]
```

```
data.isnull().sum().sum()
```

```
0
```

```
outliers = {}
for i in range(data.shape[1]):
    min_t = data[data.columns[i]].mean() - (3 * data[data.columns[i]].std())
    max_t = data[data.columns[i]].mean() + (3 * data[data.columns[i]].std())
    count = 0
    for j in data[data.columns[i]]:
        if j < min_t or j > max_t:
            count += 1
    percentage = count/data.shape[0]
    outliers[data.columns[i]] = "%.3f" % percentage

print(outliers)
```

```
{'2001': '0.019', '49.94357': '0.010', '21.47114': '0.011', '73.07750': '0.011', '8.7486
```

```
X = data.iloc[:,1:]
Y = data.iloc[:,0]
```

```
X = (X - X.mean())/X.std()
X.head()
```

	49.94357	21.47114	73.07750	8.74861	-17.40628	-13.09905	-25.01202	-12.23257	
0	0.880879	0.321962	1.763731	0.717095	-0.165521	-1.188885	0.777886	0.122554	1
1	1.251490	0.588938	1.350633	0.745954	0.000844	-0.703392	-0.066786	-0.057405	1
2	0.800152	-0.082232	0.794814	0.081837	0.336234	-1.295356	0.517344	-1.062911	-(
3	1.253665	0.794815	1.671843	0.442447	-0.411085	-0.569418	-0.712183	-0.941499	(
4	1.183286	-0.038208	2.390821	1.296056	-0.840430	-0.741985	1.612891	-0.415910	-(

5 rows × 90 columns

```
from sklearn.model_selection import train_test_split
```

```
X_shuffle = X.sample(frac=1, random_state=0)
Y_shuffle = Y.sample(frac=1, random_state=0)
```

```
x_new, x_test, y_new, y_test = train_test_split(X_shuffle, Y_shuffle, test_size=0.2, rand
dev_per = x_test.shape[0]/x_new.shape[0]
x_train, x_dev, y_train, y_dev = train_test_split(x_new, y_new, test_size=dev_per, rando
```

```
import torch
import torch.nn as nn
torch.manual_seed(0)
```

```
<torch._C.Generator at 0x7f4e43e13b58>
```

```
x_train = torch.tensor(x_train.values).float()
y_train = torch.tensor(y_train.values).float()

x_dev = torch.tensor(x_dev.values).float()
y_dev = torch.tensor(y_dev.values).float()

x_test = torch.tensor(x_test.values).float()
y_test = torch.tensor(y_test.values).float()
```

```
model = nn.Sequential(nn.Linear(x_train.shape[1],10),
                      nn.ReLU(),

                      nn.Linear(10,7),
                      nn.ReLU(),

                      nn.Linear(7,5),
                      nn.ReLU(),

                      nn.Linear(5,1))
```

```
loss_function = torch.nn.MSELoss()
```

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

```
for i in range(3000):
    y_pred = model(x_train).squeeze()
    loss = loss_function(y_pred, y_train)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

```

```
    if i%250 == 0:
        print(i, loss.item())
```

```
0 3994117.5
250 198669.84375
500 14069.74609375
750 1491.8251953125
1000 449.419677734375
1250 216.93295288085938
1500 142.8686981201172
1750 114.00255584716797
2000 101.55926513671875
2250 95.59891510009766
2500 92.46050262451172
2750 90.70108032226562
```

```
pred = model(x_test[0])
print("Ground truth:", y_test[0].item(), "Prediction:", pred.item())
```

```
Ground truth: 1998.0 Prediction: 2002.5091552734375
```