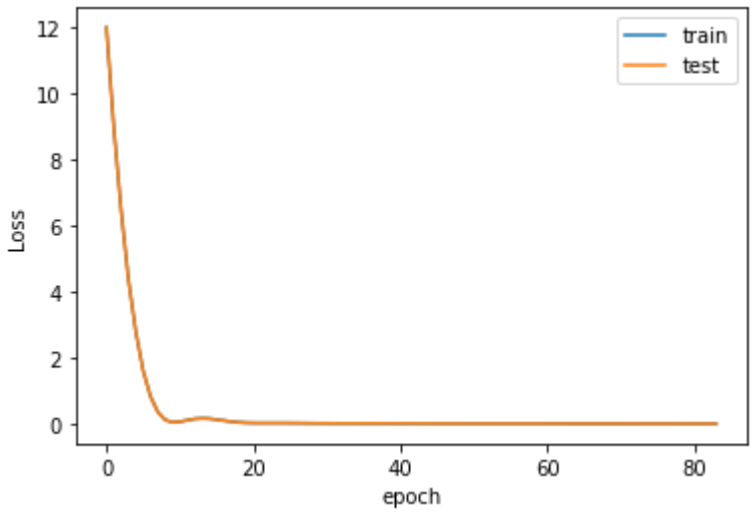
In [12]:

```
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv(r"C:\Users\adamg\Documents\ML_Research_Final_CSV.csv", header
print(f"Read in {len(df)} rows")
df1 = pd.read_csv(r"C:\Users\adamg\Documents\ML_Research_Final_CSV.csv")#, hea
df.head()
#"C:\Users\adamg\Documents\ML_Research_Final_CSV.csv"
df.replace("?", 10000, inplace=True) #10,000 is way beyond the range of column
#print(df)
fig1=sns.pairplot(data=df1)
plt.show()
df.drop([0], 0, inplace=True)
print('Data:',df)
df.head()



import numpy as np
from sklearn.model_selection import train_test_split

X_1 = np.array(df.drop([0], 1)) #last column contains label, so ignore it when
#print(f"{X}")
y_1 = np.array(df[0]) #second column is a label which is our y(Pressure)
#                                        Using the Kaggle Scaling
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pickle
import timeit
start = timeit.default_timer()
clf = MLPRegressor(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(5, 2), random_state=51)
#solver adam works for this as well
X_train, X_test, y_train, y_test = train_test_split(X_1, y_1, test_size=0.4, r
clf.fit(X_train,y_train)
clf.score(X_test,y_test)
print(f"Accuracy of MLP Regressor is:{clf.score}")
print("clf = ",clf.score)
print("X_train = ",X_train.shape)
print("X_test = ",X_test.shape)
print("y_train = ",y_train.shape)
print("y_test = ",y_test.shape)
#                                  WORKS!
#X_train, X_test, y_train, y_test = train_test_split(X_1, y_1, test_size=0.25,
nn_model = MLPRegressor(solver='adam', alpha=1e-5, hidden_layer_sizes=(100,),
#lbfgs
#y_train=y_train.astype('int')
#y_test=y_test.astype('int')
#X_test=X_test.astype('int')
#X_train=X_train.astype('int')
nn_model.fit(X_train, y_train)
nn_accuracy = nn_model.score(X_test, y_test)# Why is accuracy not used here
```

```
stop = timeit.default_timer()
prediction = nn_model.predict(X_test)
prediction=prediction.astype('float')
y_test=y_test.astype('float')
mse = mean_squared_error(y_test, prediction)
#print('Predict acc:', prediction-y_test)
print('Time:',stop-start)
print(f"Mean Squared Error is :{mse}")
#Use the mean of the y_test and prediction for mse because
#if the numbers are simply small, you can still get a low mse
# import mean squared error

nn_model.fit(X_train, y_train)            #doubt
plt.plot(nn_model.loss_curve_,label="train") #doubt
plt.plot(nn_model.loss_curve_,label="test")
plt.legend()
plt.xlabel('epoch')
plt.ylabel('Loss')
```

```
Time: 0.1231193999997231
Mean Squared Error is :0.20253208920140628
```

Out[12]: Text(0, 0.5, 'Loss')



Mse: v_x = 0.05228664057237373

Mse: v_y = 0.018790183701411427

In [ ]: