kruobzhle

January 23, 2025

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
```

[ ]:
```
dataset=pd.read_csv("byd_ev_dataset(1).csv")
dataset.head()
```

[ ]:

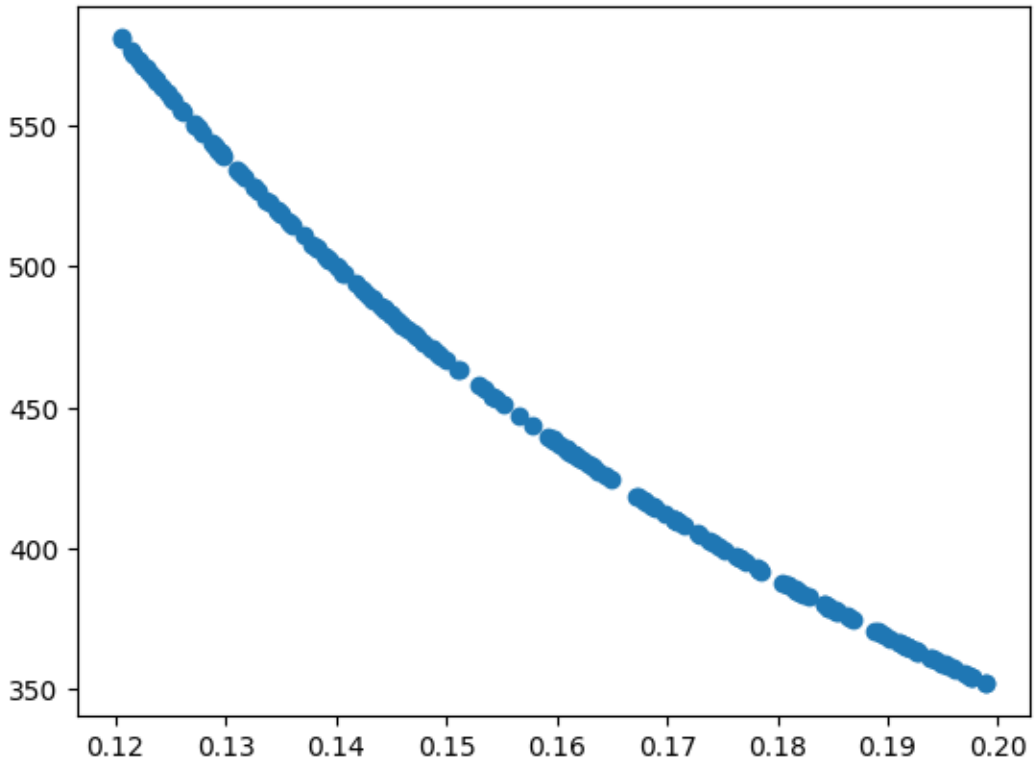
	Battery Capacity (kWh)	kWh per km	Range (km)
0	70	0.149963	466.781154
1	70	0.196057	357.038761
2	70	0.178560	392.026154
3	70	0.167893	416.933011
4	70	0.132481	528.375695

[ ]:
```

```

[ ]:
```
X=dataset['kWh per km']
y=dataset['Range (km)']
X = dataset['kWh per km'].values  # Convert to NumPy array
X = X[:, np.newaxis] # adding new axis
X.shape
```

[ ]:
```
(200, 1)
```

[ ]:
```
plt.scatter(X,y)
plt.show()
```



[ ]:
```
X_tensor=torch.tensor(X,dtype=torch.float32)
y_tensor=torch.tensor(y,dtype=torch.float32)
```

[ ]:
```
class RegressionModel(nn.Module):
  def __init__(self):
    super(RegressionModel, self).__init__()
    self.linear=nn.Linear(1,1)

  def forward(self,x):
    return self.linear(x)
```

[ ]:
```
model:RegressionModel = RegressionModel()
loss = nn.MSELoss()

criteria=torch.optim.SGD(model.parameters(), lr=0.7808342242)
```

[ ]:
```
train_losses=[]
test_losses=[]
train_accuracy=[]
train_loss=[]
```

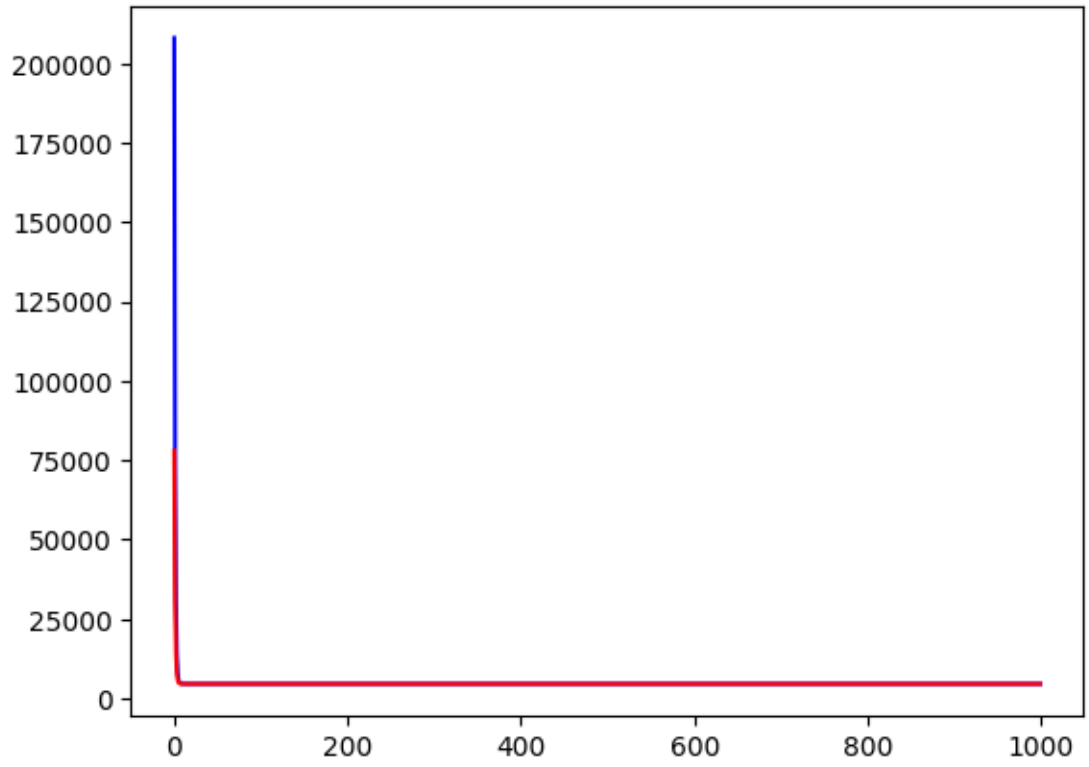
[ ]:
```
num_epochs=1000
for ep in range(num_epochs):
  model.train()
  predicted_y= model(X_tensor)
  losses=loss(y_tensor,predicted_y)
  print(losses)

  criteria.zero_grad()
  losses.backward()
  criteria.step()

  train_losses.append(losses.item())
  model.eval()

  with torch.no_grad():
    predictions=model(X_tensor)
    test_loss=loss(y_tensor,predictions)
    test_losses.append(test_loss.item())
    print(test_losses)
```

```
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:608:
UserWarning: Using a target size (torch.Size([200, 1])) that is different to the
input size (torch.Size([200])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
```

```
tensor(4571.0586, grad_fn=<MseLossBackward0>)
[78089.390625, 31130.185546875, 14166.671875, 8038.79296875, 5825.16259765625,
5025.5107421875, 4736.642578125, 4632.2890625, 4594.58984375, 4580.96826171875,
4576.04541015625, 4574.26416015625, 4573.61669921875, 4573.380859375,
4573.2919921875, 4573.25732421875, 4573.2421875, 4573.2333984375,
4573.22802734375, 4573.22216796875, 4573.2177734375, 4573.21337890625,
4573.208984375, 4573.20458984375, 4573.19970703125, 4573.1953125,
4573.1904296875, 4573.1865234375, 4573.181640625, 4573.177734375,
4573.1728515625, 4573.16845703125, 4573.1640625, 4573.1591796875,
4573.1552734375, 4573.1513671875, 4573.146484375, 4573.14208984375,
4573.13818359375, 4573.13330078125, 4573.12890625, 4573.125, 4573.12060546875,
4573.1162109375, 4573.11181640625, 4573.107421875, 4573.10302734375,
4573.0986328125, 4573.09423828125, 4573.09033203125, 4573.08544921875,
4573.0810546875, 4573.0771484375, 4573.07373046875, 4573.0693359375,
4573.064453125, 4573.06005859375, 4573.05615234375, 4573.0517578125,
4573.04736328125, 4573.04296875, 4573.0390625, 4573.03515625, 4573.03076171875,
4573.0263671875, 4573.0224609375, 4573.01806640625, 4573.01416015625,
4573.009765625, 4573.00537109375, 4573.0009765625, 4572.9970703125,
4572.99365234375, 4572.98876953125, 4572.9853515625, 4572.98095703125,
4572.9765625, 4572.97216796875, 4572.96875, 4572.96484375, 4572.96044921875,
4572.9560546875, 4572.95263671875, 4572.9482421875, 4572.94384765625,
4572.93994140625, 4572.9365234375, 4572.93212890625, 4572.92822265625,
```

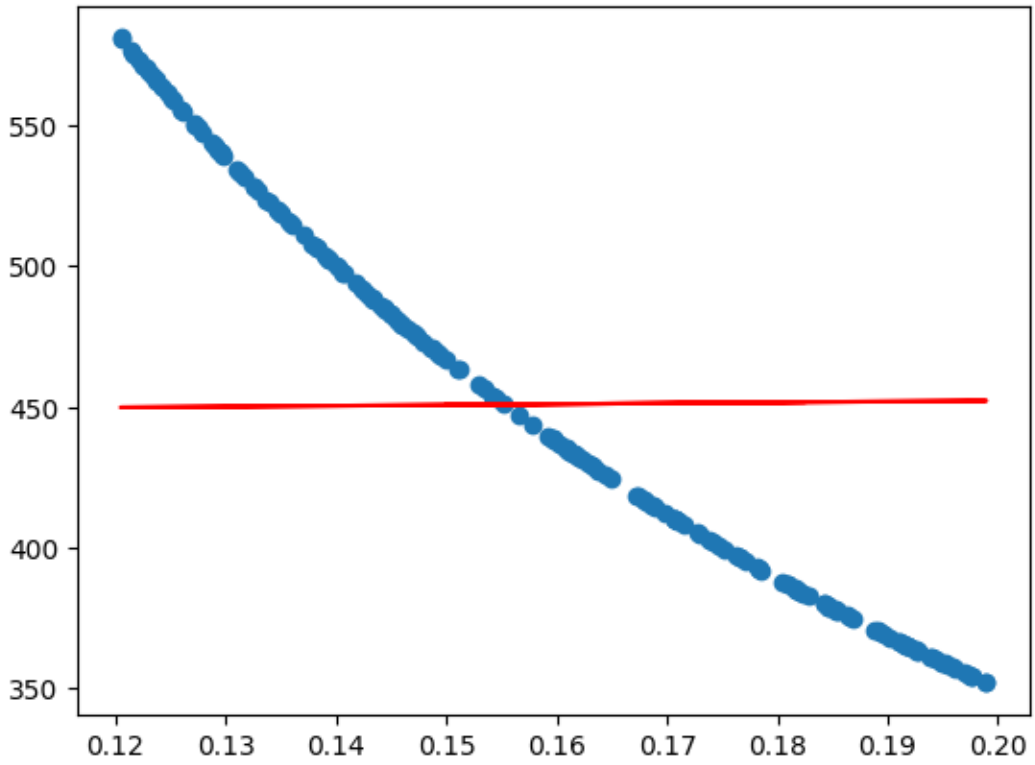
```
4570.6513671875, 4570.65087890625, 4570.65087890625, 4570.65087890625,
4570.650390625, 4570.650390625, 4570.650390625, 4570.64990234375,
4570.6494140625, 4570.6494140625, 4570.6494140625, 4570.6494140625,
4570.6494140625, 4570.6494140625, 4570.64892578125, 4570.64892578125,
4570.64892578125, 4570.64892578125, 4570.64794921875, 4570.6484375,
4570.6474609375, 4570.64697265625, 4570.64697265625, 4570.6474609375,
4570.64697265625, 4570.64697265625, 4570.64697265625, 4570.64697265625,
4570.646484375, 4570.646484375, 4570.64599609375, 4570.64599609375,
4570.646484375, 4570.6455078125, 4570.6455078125, 4570.6455078125,
4570.64501953125, 4570.64501953125, 4570.64501953125, 4570.64501953125,
4570.64501953125, 4570.64453125, 4570.64404296875, 4570.64404296875,
4570.64404296875, 4570.6435546875, 4570.6435546875, 4570.64306640625,
4570.64306640625, 4570.642578125, 4570.64306640625, 4570.642578125,
4570.642578125, 4570.642578125, 4570.64208984375, 4570.64208984375,
4570.64208984375, 4570.6416015625, 4570.64111328125]
```

[ ]:
```
plt.plot(train_losses, color='blue')
plt.plot(test_losses, color='red')
plt.show()
```



[ ]:
```
plt.scatter(X,y)
plt.plot(X,predictions, color='red')
```

```
plt.show()
```



[ ]: