```
import torch  # Core PyTorch library
import torch.nn as nn  # Neural network modules (layers, loss functions, etc.)
import torch.nn.functional as F  # Functional interface for activation functions, loss functions, etc.
from torch.utils.data import Dataset, DataLoader  # Dataset and DataLoader classes for handling data efficiently
from sklearn.model_selection import train_test_split  # Splits dataset into training and test sets

import pandas as pd  # Data manipulation and loading
import matplotlib.pyplot as plt  # Visualization library
%matplotlib inline
```

```
### Name : Sana Fathima H
### reg no : 212223240145
```

```
class Model(nn.Module):
    def __init__(self, in_features=4, h1=8, h2=9, out_features=3):
        super().__init__()
        self.fc1 = nn.Linear(in_features, h1)    # Input layer
        self.fc2 = nn.Linear(h1, h2)             # Hidden layer
        self.out = nn.Linear(h2, out_features)   # Output layer

    def forward(self, x):
        x = F.relu(self.fc1(x))  # Activation function for first hidden layer
        x = F.relu(self.fc2(x))  # Activation function for second hidden layer
        x = self.out(x)          # Output layer (no activation, raw scores)
        return x
```

```
torch.manual_seed(32)  # For reproducibility
model = Model()
```

```
df = pd.read_csv('iris.csv')
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

Next steps: Generate code with df View recommended plots New interactive sheet

```
X = df.drop('target', axis=1).values
y = df['target'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=33)
X_train = torch.FloatTensor(X_train)
X_test = torch.FloatTensor(X_test)
# y_train = F.one_hot(torch.LongTensor(y_train))  # Not needed with CrossEntropyLoss
# y_test = F.one_hot(torch.LongTensor(y_test))
y_train = torch.LongTensor(y_train)
y_test = torch.LongTensor(y_test)
```

```
trainloader = DataLoader(X_train, batch_size=60, shuffle=True)

testloader = DataLoader(X_test, batch_size=60, shuffle=False)
```

```
torch.manual_seed(4)
model = Model()
```

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

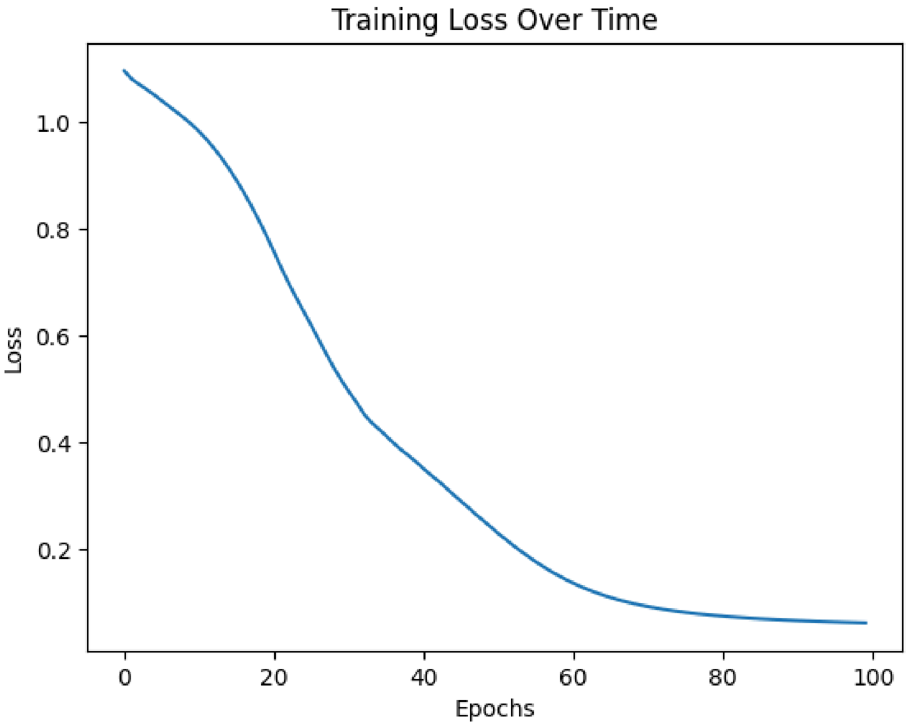
```
epochs = 100
losses = []

for i in range(epochs):
    i+=1
    y_pred = model.forward(X_train)
    loss = criterion(y_pred, y_train)
    losses.append(loss)

    # a neat trick to save screen space:
    if i%10 == 1:
        print(f'epoch: {i:2}  loss: {loss.item():10.8f}')

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

```
epoch:  1  loss: 1.09568226
epoch: 11  loss: 0.98190624
epoch: 21  loss: 0.75652373
epoch: 31  loss: 0.49447367
epoch: 41  loss: 0.34981722
epoch: 51  loss: 0.22807978
epoch: 61  loss: 0.13547550
```

```
epoch: 71  loss: 0.09162237
epoch: 81  loss: 0.07378434
epoch: 91  loss: 0.06546316
```

```
### Name : Sana Fathima H
### Reg no : 212223240145
```

```
plt.plot(range(epochs), [loss.detach().numpy() for loss in losses])
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training Loss Over Time')
plt.show()
```




```
with torch.no_grad():
    y_val = model.forward(X_test)
    loss = criterion(y_val, y_test)
print(f'{loss:.8f}')
```

```
0.06247779
```

```
correct = 0
with torch.no_grad():
    for i,data in enumerate(X_test):
        y_val = model.forward(data)
        print(f'{i+1:2}. {str(y_val):38}  {y_test[i]}')
        if y_val.argmax().item() == y_test[i]:
            correct += 1
print(f'\n{correct} out of {len(y_test)} = {100*correct/len(y_test):.2f}% correct')
```

```
 1. tensor([-0.3360,  7.3629,  1.3780])     1
 2. tensor([0.2770, 8.1552, 0.4267])        1
 3. tensor([ 11.9968,   6.1842, -19.1980])  0
 4. tensor([-2.0192,  7.9662,  4.2445])     1
 5. tensor([-6.1353,  7.9516, 11.0908])     2
 6. tensor([-10.2640,   8.3102,  17.9992])  2
 7. tensor([ 12.0541,   6.4316, -19.2913])  0
 8. tensor([ 12.9496,   6.4815, -20.7530])  0
 9. tensor([-5.7727,  8.2435, 10.5079])     2
10. tensor([-7.8872,  8.6126, 14.0726])     2
11. tensor([-8.7060,  8.6074, 15.4331])     2
12. tensor([ 11.6348,   5.8164, -18.6210])  0
13. tensor([-8.1013,  8.2331, 14.3883])     2
14. tensor([-2.0796,  7.7751,  4.3185])     1
15. tensor([-6.0833,  8.3916, 11.0582])     2
16. tensor([0.1354, 7.8658, 0.6407])        1
17. tensor([-4.0880,  7.7216,  7.6638])     2
18. tensor([ 13.1511,   6.5907, -21.0787])  0
19. tensor([-1.5649,  8.0220,  3.4751])     1
20. tensor([-6.2865,  8.9727, 11.4244])     2
21. tensor([ 12.3848,   6.2568, -19.8265])  0
22. tensor([ 13.8199,   7.0854, -22.1532])  0
23. tensor([-8.8475,  8.3181, 15.6471])     2
24. tensor([ 12.1968,   6.1261, -19.5250])  0
25. tensor([-5.8089,  7.5468, 10.5336])     2
26. tensor([-4.4530,  7.7875,  8.2861])     2
27. tensor([-1.4289,  7.7785,  3.2325])     1
28. tensor([ 0.5351,  7.5358, -0.0494])     1
29. tensor([-5.8235,  8.1573, 10.5971])     2
30. tensor([-5.2573,  7.7475,  9.6101])     2

29 out of 30 = 96.67% correct
```

```
torch.save(model.state_dict(), 'IrisDatasetModel.pt')
```

```
new_model = Model()
new_model.load_state_dict(torch.load('IrisDatasetModel.pt'))
new_model.eval()
```

```
Model(
  (fc1): Linear(in_features=4, out_features=8, bias=True)
  (fc2): Linear(in_features=8, out_features=9, bias=True)
```

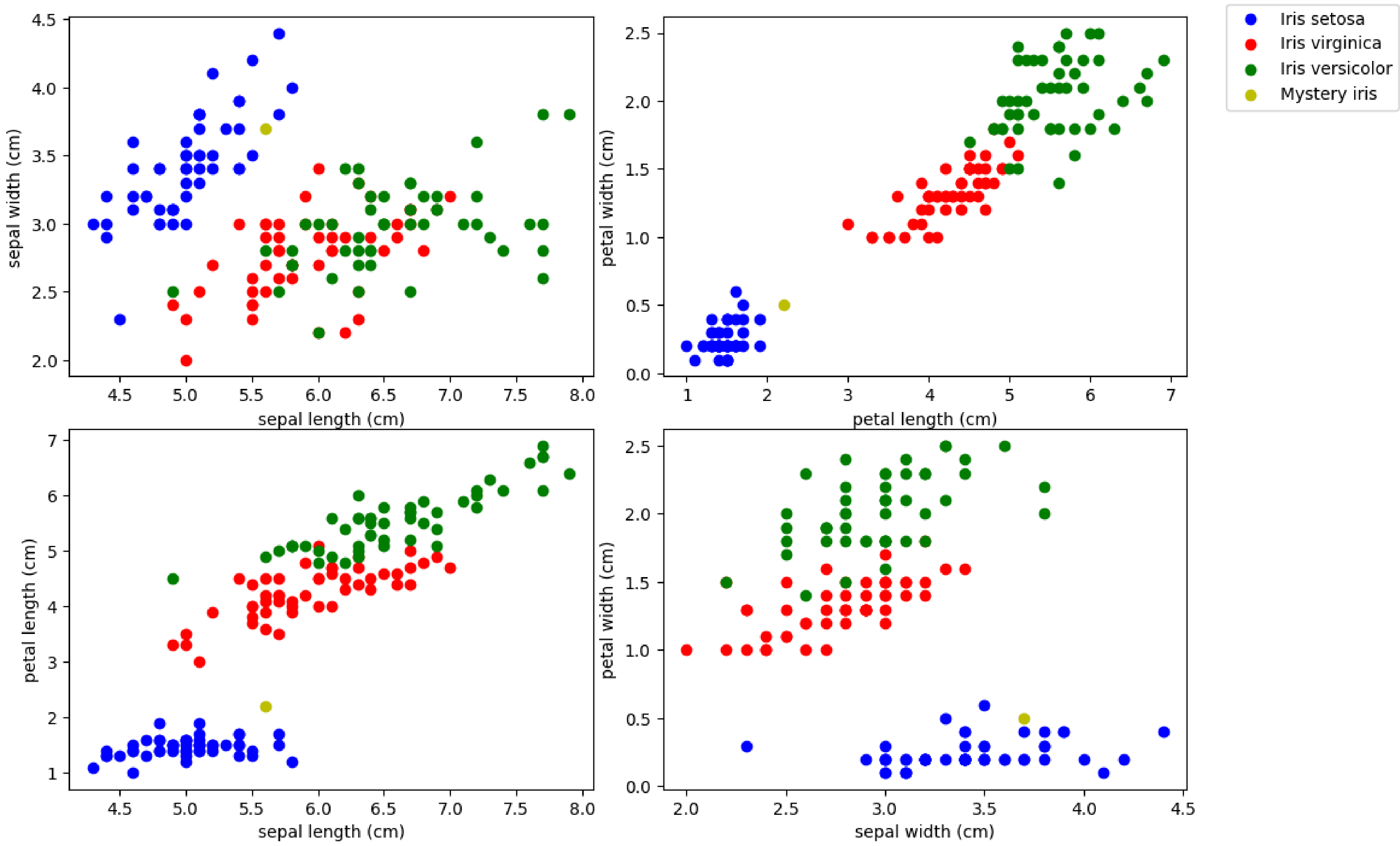
```
    (out): Linear(in_features=9, out_features=3, bias=True)
  )
```

```
with torch.no_grad():
    y_val = new_model.forward(X_test)
    loss = criterion(y_val, y_test)
print(f'{loss:.8f}')
```

```
0.06247779
```

```
mystery_iris = torch.tensor([5.6,3.7,2.2,0.5])
```

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10,7))
fig.tight_layout()

plots = [(0,1),(2,3),(0,2),(1,3)]
colors = ['b', 'r', 'g']
labels = ['Iris setosa','Iris virginica','Iris versicolor','Mystery iris']

for i, ax in enumerate(axes.flat):
    for j in range(3):
        x = df.columns[plots[i][0]]
        y = df.columns[plots[i][1]]
        ax.scatter(df[df['target']==j][x], df[df['target']==j][y], color=colors[j])
        ax.set(xlabel=x, ylabel=y)

    # Add a plot for our mystery iris:
    ax.scatter(mystery_iris[plots[i][0]],mystery_iris[plots[i][1]], color='y')

fig.legend(labels=labels, loc=3, bbox_to_anchor=(1.0,0.85))
plt.show()
```




```
with torch.no_grad():
    print(new_model(mystery_iris))
    print()
    print(labels[new_model(mystery_iris).argmax()])
```

```
tensor([ 12.2112,   7.1279, -19.5248])

Iris setosa
```

```
### Name : Sana Fathima H
### reg no : 212223240145
```