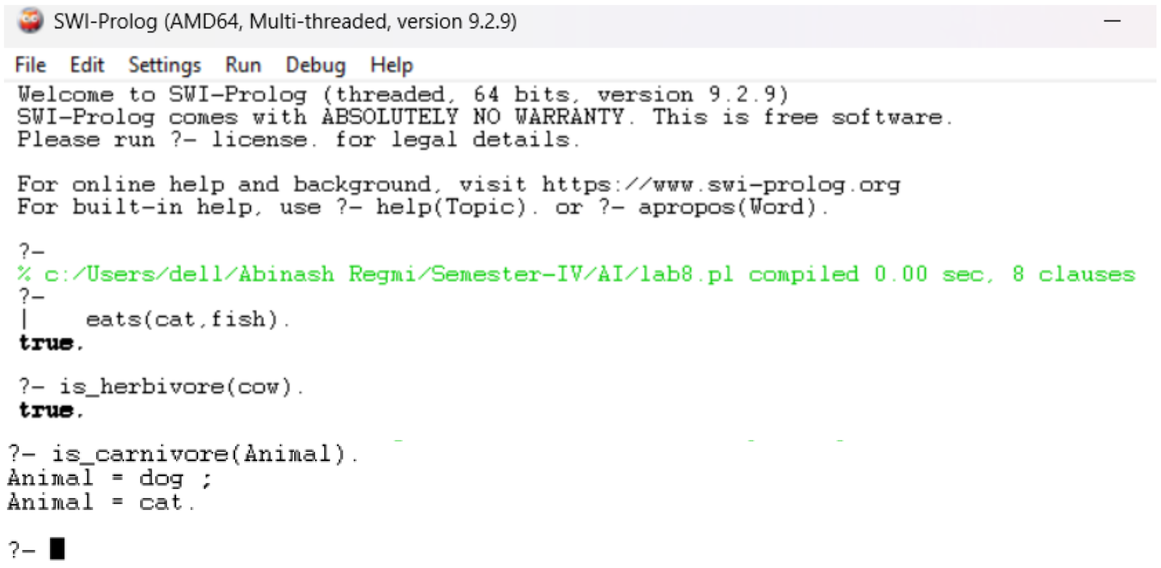
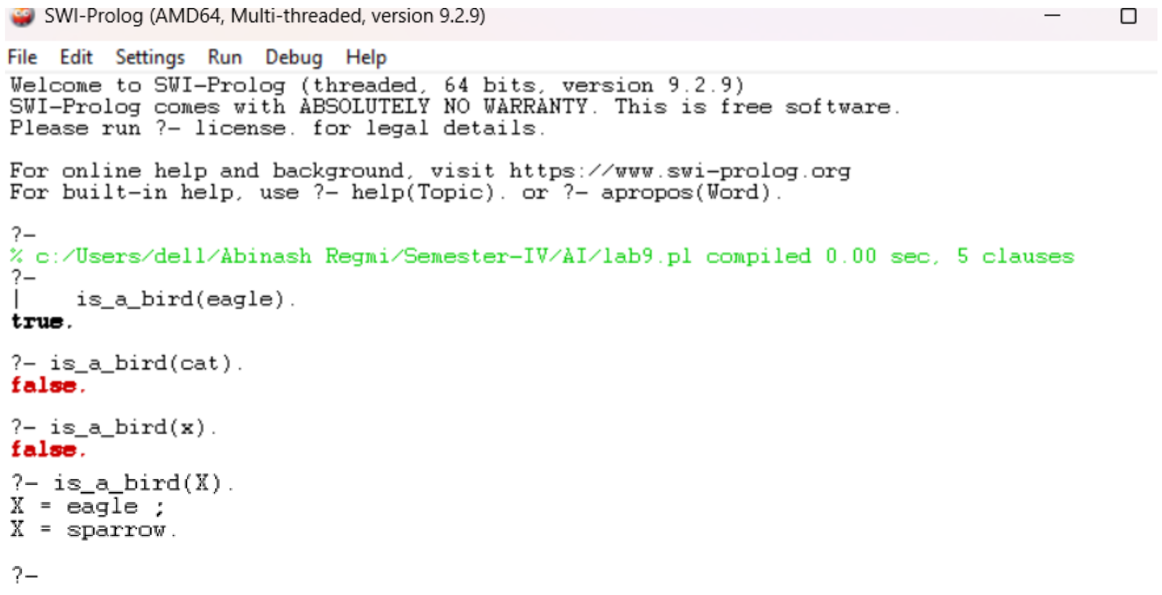
OUTPUT:


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

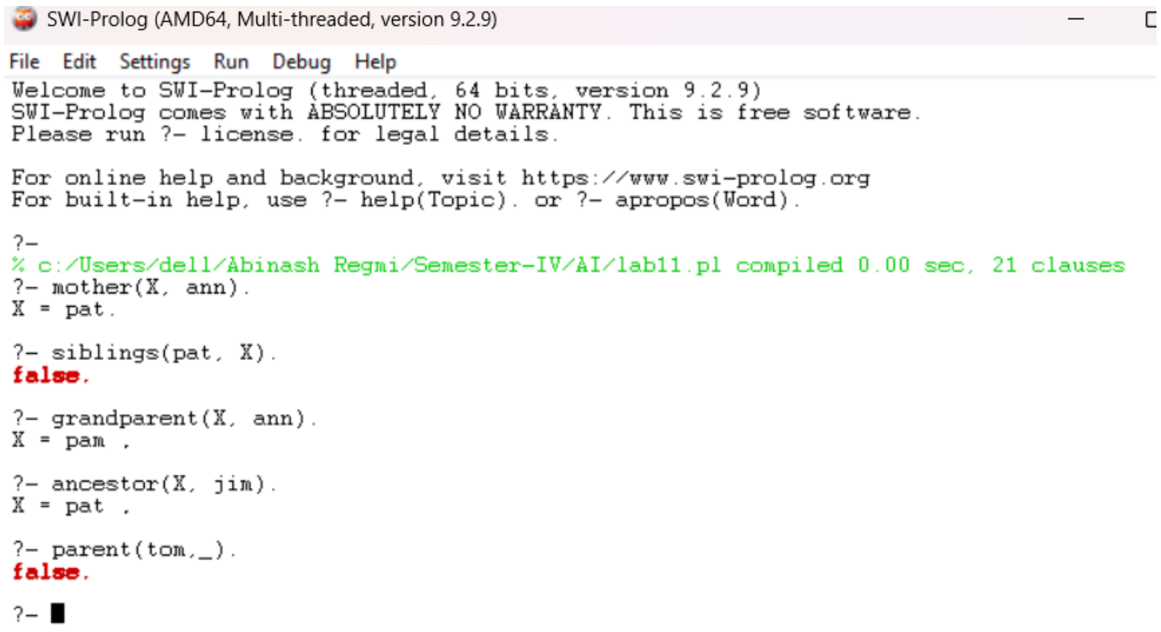
```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/dell/Abinash Regmi/Semester-IV/AI/lab8.pl compiled 0.00 sec, 8 clauses
?-
|    eats(cat,fish).
true.

?- is_herbivore(cow).
true.

?- is_carnivore(Animal).
Animal = dog ;
Animal = cat.

?-
```

OUTPUT:


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

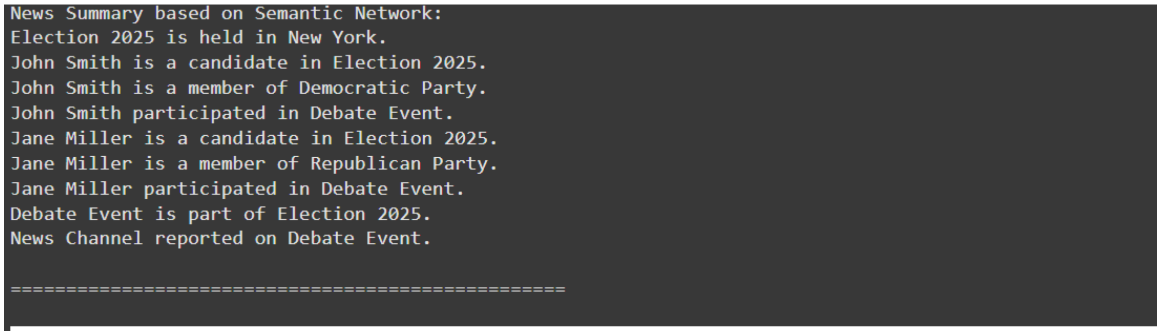
```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/dell/Abinash Regmi/Semester-IV/AI/lab9.pl compiled 0.00 sec, 5 clauses
?-
|    is_a_bird(eagle).
true.

?- is_a_bird(cat).
false.

?- is_a_bird(x).
false.

?- is_a_bird(X).
X = eagle ;
X = sparrow.

?-
```

OUTPUT:

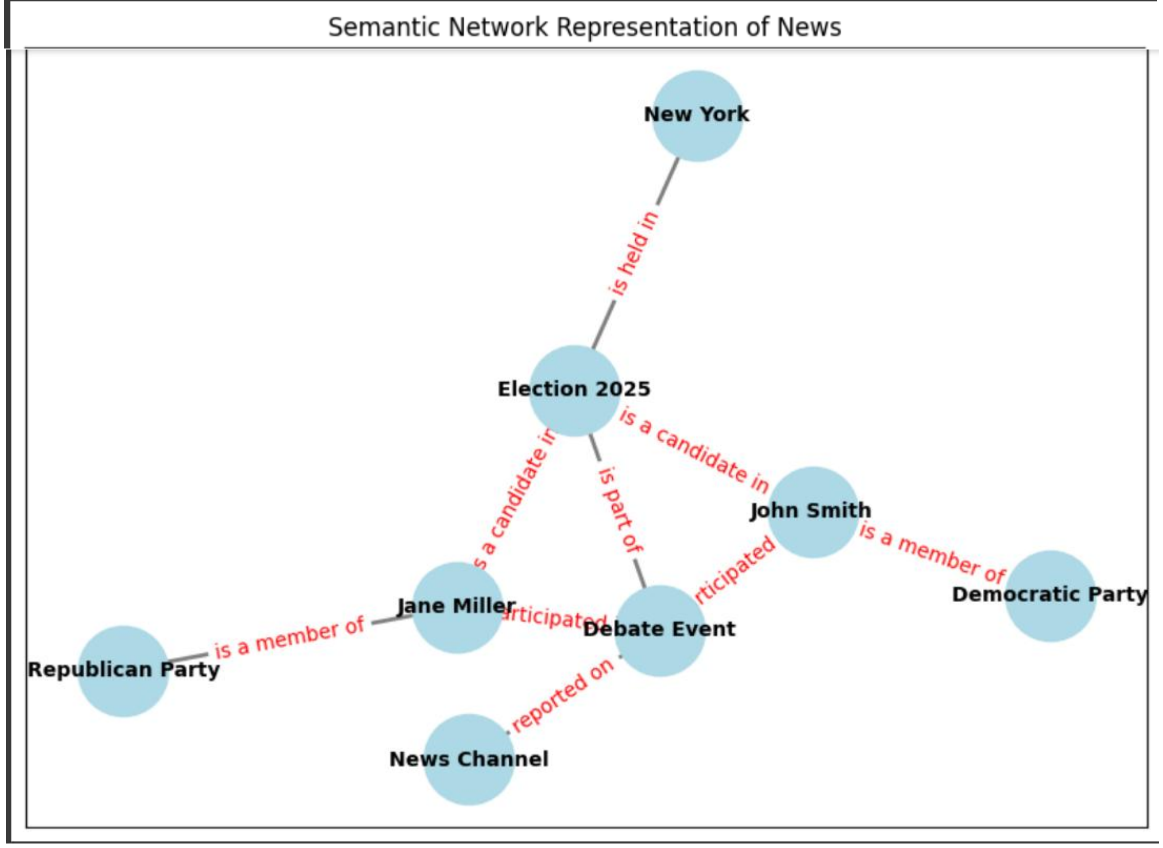

SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/dell/Abinash Regmi/Semester-IV/AI/lab11.pl compiled 0.00 sec, 21 clauses
?- mother(X, ann).
X = pat.

?- siblings(pat, X).
false.

?- grandparent(X, ann).
X = pam .

?- ancestor(X, jim).
X = pat .

?- parent(tom,_).
false.

?-
```

OUTPUT:



```
News Summary based on Semantic Network:
Election 2025 is held in New York.
John Smith is a candidate in Election 2025.
John Smith is a member of Democratic Party.
John Smith participated in Debate Event.
Jane Miller is a candidate in Election 2025.
Jane Miller is a member of Republican Party.
Jane Miller participated in Debate Event.
Debate Event is part of Election 2025.
News Channel reported on Debate Event.

==================================================
```



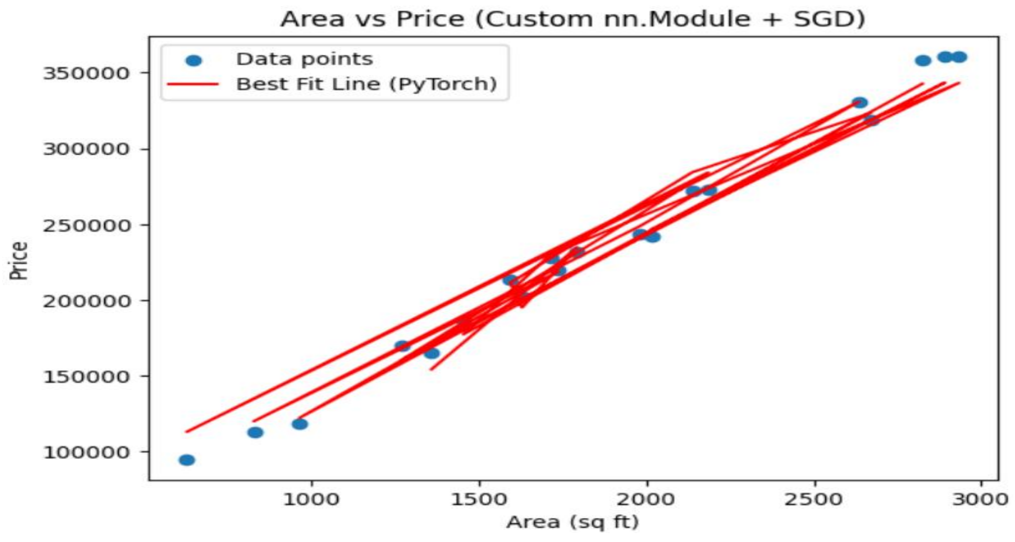
SOURCE CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('housing_prices.csv')

X = df[['area_sqft', 'num_rooms']].values
y = df['price'].values

X = (X - X.mean(axis=0)) / X.std(axis=0)

y_min, y_max = y.min(), y.max()
y_norm = (y - y_min) / (y_max - y_min)

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

weights = np.random.randn(2)
bias = 0

lr = 0.1
epochs = 1000

for epoch in range(epochs):
    z = np.dot(X, weights) + bias
    y_pred = sigmoid(z)

    mse_loss = np.mean((y_pred - y_norm) ** 2)
    if((epoch+1)%50==0):
      print(f"Epoch:{epoch+1}, loss= {mse_loss}")

    dE_dy = (y_pred - y_norm)
    dy_dz = sigmoid(z) * (1 - sigmoid(z))
    dz_dw = X
    dz_db = 1
    grad_w = np.dot(dz_dw.T, dE_dy * dy_dz) / len(X)
    grad_b = np.sum(dE_dy * dy_dz * dz_db) / len(X)

    weights -= lr * grad_w
    bias -= lr * grad_b

weights

bias

test_area = 2135
test_rooms = 4

test_X = np.array([test_area, test_rooms])
test_X_norm = (test_X - df[['area_sqft', 'num_rooms']].mean().values) / df[['area_sqft',
    'num_rooms']].std().values

z_test = np.dot(test_X_norm, weights) + bias
pred_norm = sigmoid(z_test)

pred_price = pred_norm * (y_max - y_min) + y_min

print(f"Predicted Price for area={test_area} sq ft and rooms={test_rooms}:
    {pred_price:.2f}")

import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt

df = pd.read_csv('housing_prices.csv')

X = df[['area_sqft', 'num_rooms']].values
y = df['price'].values
```

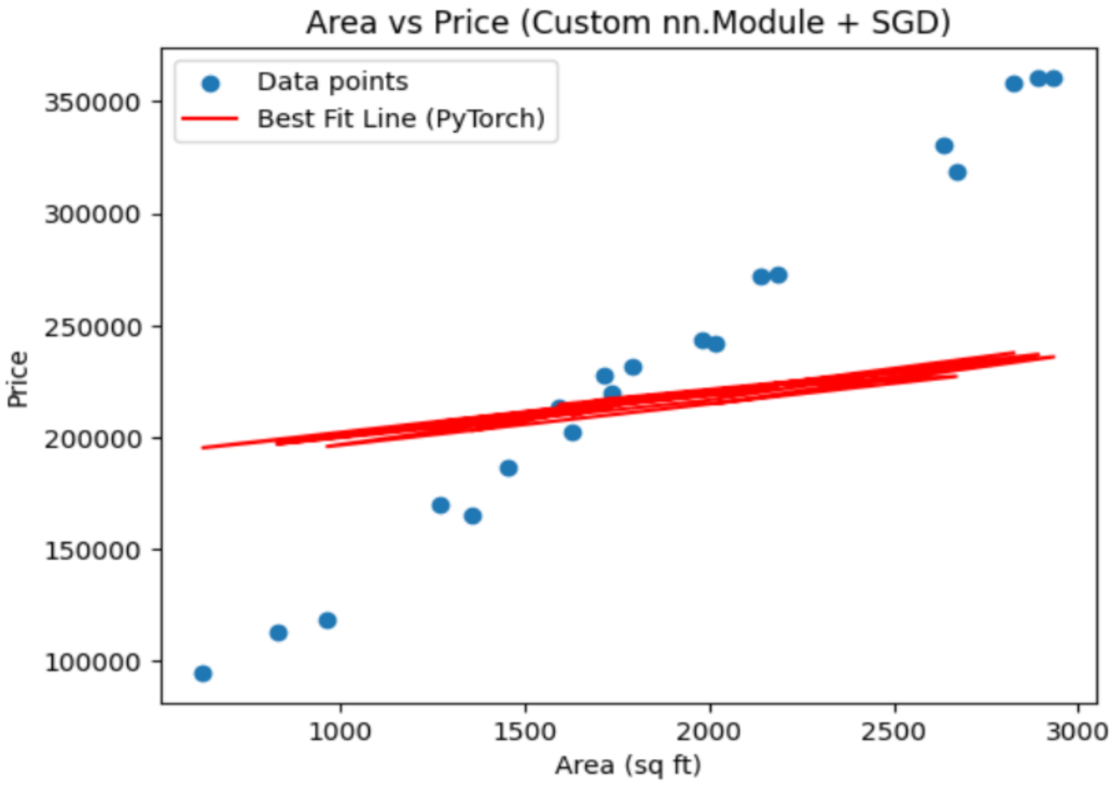
```
X_mean = X.mean(axis=0)
X_std = X.std(axis=0)
X_norm = (X - X_mean) / X_std

y_min, y_max = y.min(), y.max()
y_norm = (y - y_min) / (y_max - y_min)

X_tensor = torch.tensor(X_norm, 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(2, 1)

    def forward(self, x):
        z = self.linear(x)
        return torch.sigmoid(z)

model = SingleLayerModel()

criterion = nn.MSELoss()

optimizer = optim.SGD(model.parameters(), lr=0.01)

epochs = 10000
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()

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.scatter(df['area_sqft'], df['price'], label="Data points")
plt.plot(df['area_sqft'], final_preds_denorm, color='red', label="Best Fit Line
    (PyTorch)")
plt.xlabel("Area (sq ft)")
plt.ylabel("Price")
plt.title("Area vs Price (Custom nn.Module + SGD)")
plt.legend()
plt.show()

for name, param in model.named_parameters():
    print(f"{name}: {param.data.numpy()}")
print("Final MSE Loss:", loss.item())

test_input = np.array([[2135, 4]], dtype=np.float32)
test_input_norm = (test_input - X_mean) / X_std
test_tensor = torch.tensor(test_input_norm, dtype=torch.float32)

with torch.no_grad():
    pred_norm = model(test_tensor).item()

pred_price = pred_norm * (y_max - y_min) + y_min
print(f"Predicted Price for area=2135 sq ft and rooms=4: {pred_price:.2f}")
```

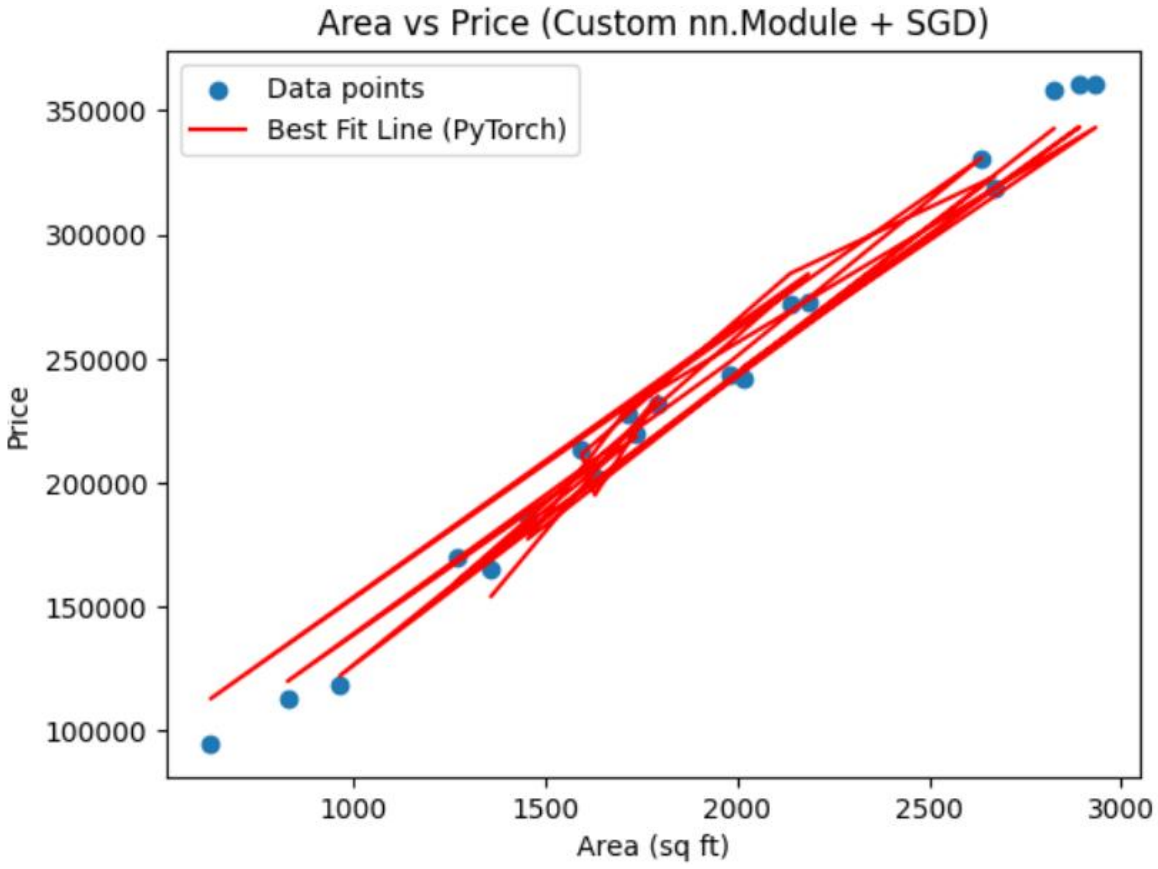
OUTPUT (lr=0.1)**:**



OUTPUT (lr=0.001):



OUTPUT (lr=0.4):


Area vs Price (Custom nn.Module + SGD)
Data points
Best Fit Line (PyTorch)
Price
Area (sq ft)