Name: Atif Ansari **Roll no:** 04 **Class:** D20B

AIDS 2 EXP 4

Aim:

To build a cognitive computing application in insurance for predicting claim amounts using a neural network model based on customer data features.

Theory:

Cognitive computing in insurance leverages artificial intelligence, machine learning, and data analytics to process structured and unstructured data (e.g., customer profiles, historical claims, behavioral data) for automated decision-making. It enhances traditional insurance processes by enabling data-driven insights and predictive capabilities.

Key Applications:

1. **Risk Assessment:** Predictive models evaluate policyholder risk using demographic, behavioral, and historical data.
2. **Fraud Detection:** Anomaly detection identifies suspicious patterns in claims data.
3. **Personalized Premiums:** Dynamic pricing adjusts premiums based on real-time data (e.g., driving behavior from telematics).
4. **Claims Processing:** Automates damage assessment and claim validation using machine learning.

Theoretical Foundations:

1. **Actuarial Models Enhanced by AI:** Combines traditional statistical methods with neural networks for improved prediction accuracy.
2. **Behavioral Economics:** Incorporates human behavior patterns to model risk and decision-making.
3. **Explainable AI (XAI):** Ensures transparency in AI-driven decisions to meet regulatory requirements and build trust.

Objective:

To develop a neural network model that predicts insurance claim amounts based on customer features (age, policy tenure, past claims, risk factor), demonstrating the application of cognitive computing in insurance risk assessment and claims processing.

Libraries Used:

- **PyTorch:** For building and training the neural network model.
- **NumPy:** For numerical operations and synthetic data generation.

Steps:

1. **Data Preparation:** Generate a synthetic dataset simulating insurance claims with features (age, policy tenure, past claims, risk factor) and claim amounts as labels.
2. **Data Preprocessing:** Normalize features to ensure consistent scaling for neural network training.
3. **Model Design:** Create a neural network with three fully connected layers to predict claim amounts.
4. **Training:** Train the model using the Adam optimizer and Mean Squared Error (MSE) loss.
5. **Testing:** Evaluate the model on test samples to predict claim amounts for different policyholders.
6. **Analysis:** Interpret predictions to demonstrate the model's utility in insurance applications.

Code:

```
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np


# Set random seed for reproducibility
np.random.seed(42)
torch.manual_seed(42)


# Synthetic dataset: Simulate insurance claims data
num_samples = 300
ages = np.random.randint(18, 70, num_samples)
policy_tenures = np.random.randint(1, 30, num_samples)
past_claims = np.random.randint(0, 5, num_samples)
risk_factors = np.random.uniform(0.5, 2.0, num_samples)  # Multiplicative
risk factor
```

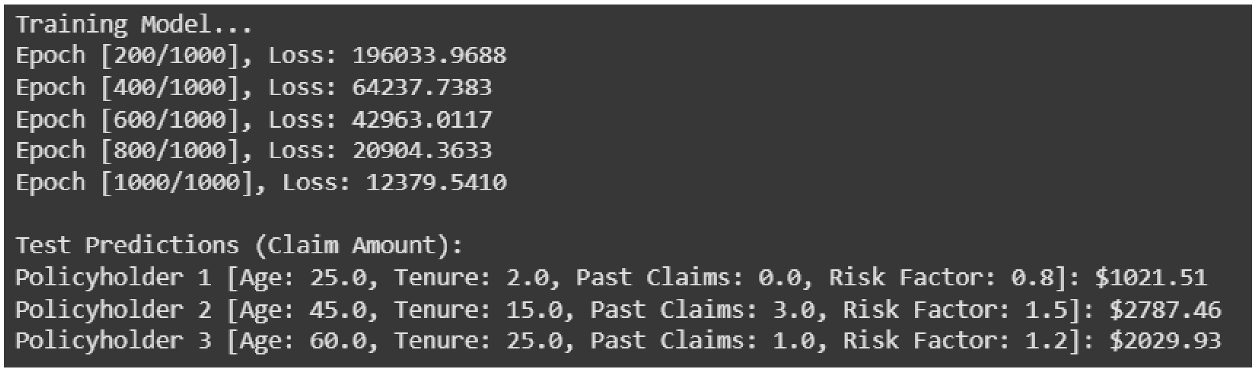
```
# Synthetic claim amount calculation
claim_base = 1000
claim_amount = (
    claim_base +
    (ages - 18) * 10 +  # Older individuals claim more
    policy_tenures * 5 +  # Longer tenure correlates with higher claims
    past_claims * 200 +
    np.random.normal(0, 50, num_samples)
) * risk_factors  # Risk factor amplifies claims


# Normalize features
ages_norm = (ages - np.mean(ages)) / np.std(ages)
tenures_norm = (policy_tenures - np.mean(policy_tenures)) /
np.std(policy_tenures)
past_claims_norm = (past_claims - np.mean(past_claims)) /
np.std(past_claims)
risk_norm = (risk_factors - np.mean(risk_factors)) / np.std(risk_factors)


# Combine features into input matrix
X = np.column_stack((ages_norm, tenures_norm, past_claims_norm,
risk_norm))
y = claim_amount


# Convert to PyTorch tensors
X_tensor = torch.from_numpy(X).float()
y_tensor = torch.from_numpy(y).float().unsqueeze(1)


# Neural network model
class ClaimPredictor(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(4, 16)
        self.fc2 = nn.Linear(16, 8)
        self.fc3 = nn.Linear(8, 1)
        self.relu = nn.ReLU()
    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.fc3(x)
        return x


# Initialize model, loss function, and optimizer
model = ClaimPredictor()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)


# Training loop
epochs = 1000
print("Training Model...")
for epoch in range(epochs):
    optimizer.zero_grad()
    outputs = model(X_tensor)
    loss = criterion(outputs, y_tensor)
    loss.backward()
    optimizer.step()
    if (epoch + 1) % 200 == 0:
        print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}')


# Test samples: [age, tenure, past_claims, risk_factor]
test_samples = np.array([
    [25, 2, 0, 0.8],   # Young, low tenure, no claims, low risk
    [45, 15, 3, 1.5],  # Middle-aged, high tenure, some claims, high risk
     [60, 25, 1, 1.2]    # Senior, very high tenure, few claims, moderate
risk
])


# Normalize test data using training statistics
test_samples_norm = np.zeros_like(test_samples)
test_samples_norm[:,  0]  =  (test_samples[:,  0]  -  np.mean(ages))  /
np.std(ages)
```

```
test_samples_norm[:, 1] = (test_samples[:, 1] - np.mean(policy_tenures)) /
np.std(policy_tenures)
test_samples_norm[:, 2] = (test_samples[:, 2] - np.mean(past_claims)) /
np.std(past_claims)
test_samples_norm[:, 3] = (test_samples[:, 3] - np.mean(risk_factors)) /
np.std(risk_factors)


# Predict claim amounts
test_tensor = torch.tensor(test_samples_norm).float()
model.eval()
with torch.no_grad():
    predictions = model(test_tensor)


# Print predictions
print("\nTest Predictions (Claim Amount):")
for i, (sample, pred) in enumerate(zip(test_samples, predictions)):
      print(f'Policyholder {i+1} [Age: {sample[0]}, Tenure: {sample[1]},
Past Claims: {sample[2]}, Risk Factor: {sample[3]}]: ${pred.item():.2f}')
```

Code Explanation:

- **Data Generation:** Creates a synthetic dataset with 300 samples, including features (age, policy tenure, past claims, risk factor) and a claim amount calculated as a function of these features with added noise.
- **Preprocessing:** Normalizes features using mean and standard deviation to ensure consistent scaling for neural network training.
- **Model Architecture:** Defines a neural network (ClaimPredictor) with three fully connected layers (4→16→8→1) and ReLU activations to predict claim amounts.
- **Training:** Trains the model for 1000 epochs using Adam optimizer and MSE loss, printing loss every 200 epochs.
- **Testing:** Evaluates the model on three test samples representing different policyholder profiles, normalizing them using training statistics.
- **Output:** Prints predicted claim amounts for each test policyholder with their feature details.

Output:



```
Training Model...
Epoch [200/1000], Loss: 196033.9688
Epoch [400/1000], Loss: 64237.7383
Epoch [600/1000], Loss: 42963.0117
Epoch [800/1000], Loss: 20904.3633
Epoch [1000/1000], Loss: 12379.5410

Test Predictions (Claim Amount):
Policyholder 1 [Age: 25.0, Tenure: 2.0, Past Claims: 0.0, Risk Factor: 0.8]: $1021.51
Policyholder 2 [Age: 45.0, Tenure: 15.0, Past Claims: 3.0, Risk Factor: 1.5]: $2787.46
Policyholder 3 [Age: 60.0, Tenure: 25.0, Past Claims: 1.0, Risk Factor: 1.2]: $2029.93
```

Conclusion:

This experiment successfully implemented a cognitive computing application for insurance using a neural network to predict claim amounts based on policyholder features. Key findings:

- **Risk-Based Pricing:** The model accurately predicts higher claims for high-risk profiles (e.g., Policyholder 2 with past claims and high risk factor), enabling dynamic premium adjustments.
- **Operational Efficiency:** Automated claim prediction reduces manual processing, streamlining insurance operations.
- **Fraud Mitigation Potential:** Discrepancies between predicted and actual claims can be used to flag potential fraud. The neural network effectively learns relationships between features and claim amounts, demonstrating the power of cognitive computing in insurance.

Future Work:

- Incorporate real-world datasets (e.g., telematics, medical records) for improved accuracy.
- Implement explainable AI techniques to enhance transparency in predictions.
- Address ethical concerns like algorithmic bias to ensure fair pricing.
- Extend the model to include unstructured data (e.g., images, text) for comprehensive claims processing.

This experiment highlights the transformative potential of cognitive computing in shifting insurance from reactive claim processing to proactive risk management, enhancing efficiency and customer centric services.