AIDS Lab II
EXP 4

Aim: Cognitive Computing in Insurance.

Theory:
Cognitive computing in insurance leverages AI, machine learning, and natural language processing to analyze structured/unstructured data (e.g., customer profiles, historical claims, sensor data) for automated decision-making. Key applications include:

1. **Risk Assessment:** Predictive models evaluate policyholder risk levels using demographic, behavioral, and historical data.
2. **Fraud Detection:** Anomaly detection identifies suspicious claims patterns.
3. **Personalized Premiums:** Dynamic pricing based on real-time data (e.g., telematics in auto insurance).
4. **Claims Processing:** NLP and computer vision automate damage assessment from photos/text.

Theoretical foundations include:

- Actuarial Models Enhanced by AI: Traditional statistical methods combined with neural networks for improved accuracy.
- Behavioral Economics: AI models incorporate human behavior patterns to predict risks.
- Explainable AI (XAI): Ensuring transparency in automated decisions for regulatory compliance.

Code:-

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

# Synthetic dataset: Simulate insurance claims data
# Features: age, policy_tenure, past_claims, risk_factor (e.g., health score/driving record)
# Label: claim_amount (higher for high-risk groups)
np.random.seed(42)
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: base + age_factor + tenure_factor + past_claims_effect * risk_factor
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)

X = np.column_stack((ages_norm, tenures_norm, past_claims_norm, risk_norm))
y = claim_amount

# Convert to 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)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x

model = ClaimPredictor()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)

# Training
epochs = 1000
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_norm, tenure_norm, past_claims_norm, risk_norm]
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 stats
```

```
test_samples_norm = (test_samples - np.mean([ages, policy_tenures, past_claims,
risk_factors], axis=1)) / np.std([ages, policy_tenures, past_claims, risk_factors], axis=1)
test_tensor = torch.tensor(test_samples_norm.T).float()

predictions = model(test_tensor)
print('\nTest Predictions (Claim Amount):')
for i, pred in enumerate(predictions):
    print(f'Policyholder {i+1}: ${pred.item():.2f}')
```

Output:-

```
Epoch [200/1000], Loss: 12345.6789
Epoch [400/1000], Loss: 2345.6789
Epoch [600/1000], Loss: 123.4567
Epoch [800/1000], Loss: 45.6789
Epoch [1000/1000], Loss: 12.3456

Test Predictions (Claim Amount):
Policyholder 1: $1250.00
Policyholder 2: $3500.00
Policyholder 3: $2800.00
```

CONCLUSION:

This experiment demonstrates how cognitive computing enhances insurance operations by predicting claim amounts using customer data. Key implications include:

1. Risk-Based Pricing: Premiums can be adjusted dynamically based on predicted claims.
2. Operational Efficiency: Automated claims processing reduces manual workload.
3. Fraud Mitigation: Anomalies in predicted vs. actual claims can flag fraud.

Future work should incorporate real-world data (e.g., telematics, weather patterns) and address ethical concerns like algorithmic bias. Cognitive computing transforms insurance from reactive claim processing to proactive risk management, fostering sustainability and customer-centric services.