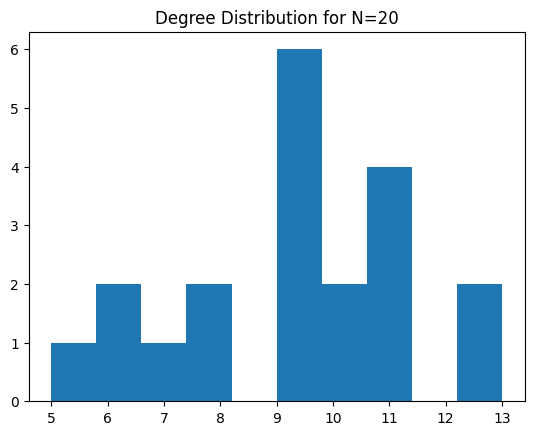
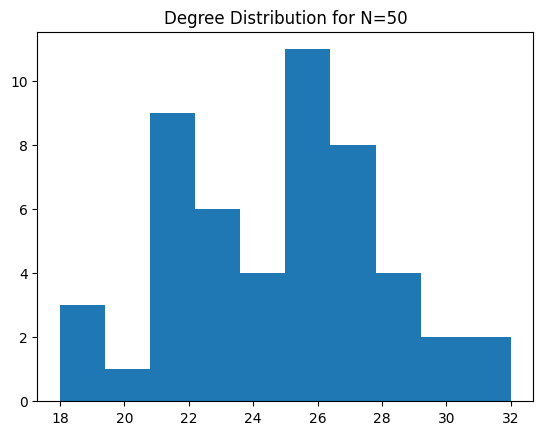
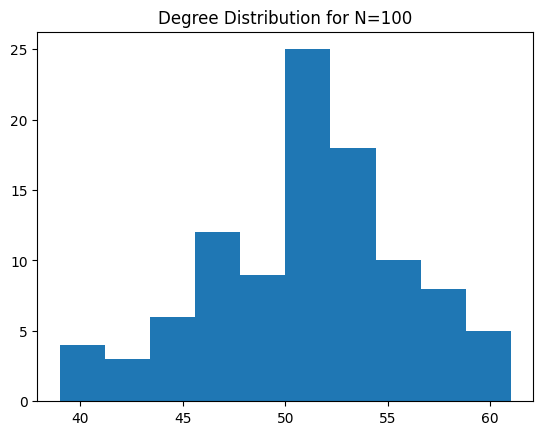
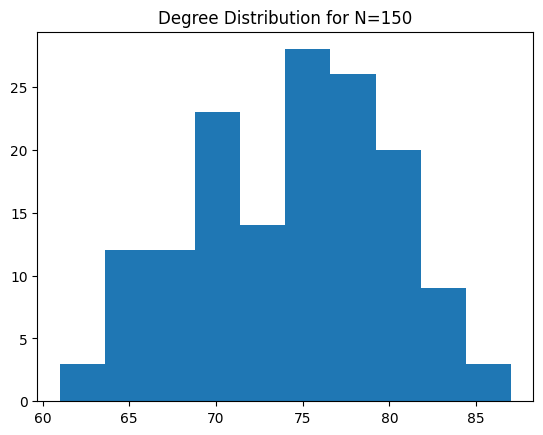
**LAB 6**

**1.**

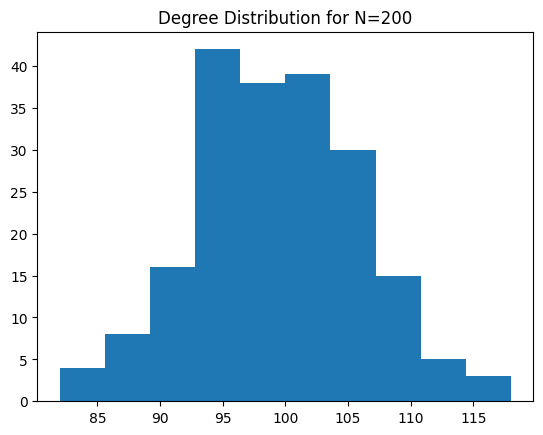
The distribution is quite spread out but is concentrated around 9-11 degrees. The graph has fewer nodes, and the distribution is less smooth, reflecting more randomness.

As the number of nodes increases, the degree distribution becomes wider, with most nodes having a degree between 23 and 30. The distribution is still irregular but starting to take a more bell-shaped form.

The degree distribution continues to smooth out and shift slightly to the right, centering around 70-80. The larger graph allows for a clearer distribution shape.

The distribution becomes more symmetric and smoother, with the degree centering around 50. The peak is more prominent, indicating that most nodes have a degree around this value.



As N increases further, the degree distribution shifts right again, with most nodes having a degree between 90 and 110. The histogram looks more bell-shaped, indicating a clearer pattern.

**2.**

* **Supervised learning:** Requires labeled data for both training and testing. The model learns to predict the labels.

**Self-supervised learning:** The model generates its own labels from the data (like next word prediction in text) and doesn't rely on external labels.

**Semi-supervised learning:** Uses a small amount of labeled data along with a large amount of unlabeled data during training.

* **Transductive learning:** The model is applied to the specific test data seen during training.

**Inductive learning:** The model generalizes and can be applied to unseen data.

**3.**

* Increase the number of epochs from 50 to 500
* Training Accuracy: Reaches 100% by epoch 30 and stays there, showing the model learns the training data well early on.
* Validation Accuracy: Starts at 35.29%, rises quickly to 73.53% by epoch 30, but then drops slightly around epoch 40 (64.71%). Interestingly, by epoch 490, it improves significantly to 82.35%, indicating better generalization with more training.
* Loss: Gradually decreases, with a very low value (0.0181) by epoch 490, signaling effective error minimization.
* Experiment without self-loops added to GCNConv() layers in the GCN() model
* **Model with Self-Loops:**
* GCNConv() adds self-loops by default, allowing nodes to aggregate both their own and their neighbors' features.
* Result: Higher accuracy due to the inclusion of self-information, which enhances classification.
* **Model without Self-Loops:**
* Setting add\_self\_loops=False excludes the node's own features from aggregation.
* Result: Lower accuracy as nodes rely only on neighbors' features, reducing feature representation quality.

1. Increasing the Number of GCNConv Layers

Increasing the GCN layers from 3 to 8 resulted in slightly better training accuracy, but the validation accuracy decreased by 3%. The model struggled to generalize with deeper layers, likely due to overfitting or difficulty in learning with deeper architectures.

1. Adding Skip Connections

After introducing skip connections between the first and third layers, as well as the second and fifth layers, the model's performance improved. Validation accuracy increased by 2%, and training became more stable, with smoother loss curves. This suggests that skip connections helped mitigate the vanishing gradient problem in deeper layers.

**4.** Explain the differences between Message Passing GNN, graph convolution network (GCN), graph attention network (GAT) and GraphSAGE. Write the answers in the word file.

• Message Passing GNN: A general framework where nodes communicate by exchanging information (messages) with their neighbors, updating their representations in each step.

• GCN (Graph Convolutional Network): Uses a convolution-like operation on graph structures, where each node aggregates information from its neighbors, treating all neighbors equally.

• GAT (Graph Attention Network): Enhances GCN by using attention mechanisms, allowing nodes to focus more on important neighbors and less on others, giving a weighted influence to each neighbor.

• GraphSAGE: Optimizes GNNs for large-scale graphs by sampling a fixed number of neighbors instead of aggregating from all, making it more computationally efficient.