GRAPH CLASSIFICATION USING GRAPH NEURAL NETWORK

A Project Report Submitted

in Partial Fulfillment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

Sulthan Vishnu Sai

(Roll No. CS19b1025)

ಭಾರತೀಯ ಮಾಹಿತಿ ತಂತ್ರಜ್ಞಾನ ಸಂಸ್ಥೆ ರಾಯಚೂರು
भारतीय सूचना प्रौद्योगिकी संस्थान रायचूर
Indian Institute of Information Technology Raichur

To

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY RAICHUR-784135, INDIA

May 2023

DECLARATION

I, **Sulthan Vishnu Sai (Roll No: CS19B1025)**, hereby declare that, this report entitled **"Graph Classification Using Graph Neural Network"** submitted to Indian Institute of Information Technology Raichur towards partial requirement of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** is an original work carried out by me under the supervision of **Priodyuti Pradhan** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever an external information or statement or result is used then, that has been duly acknowledged and cited.

Raichur-584135

9 May 2023

Sulthan Vishnu Sai

CERTIFICATE

This is to certify that the work contained in this project report entitled **"Graph Classification Using Graph Neural Network"** submitted by **Sulthan Vishnu Sai** (**Roll No: CS19B1025**) to the Indian Institute of Information Technology Raichur towards partial requirement of **Bachelor of Technology** in **Computer Science and Engineering** has been carried out by [him/her] under my supervision and that it has not been submitted elsewhere for the award of any degree.

Raichur-584135 **Priodyuti Pradhan**

9 May 2023

ABSTRACT

Graph Classification using Graph Neural Network

Supervised learning on Graph Neural Networks (GNNs) have established themselves as the advanced models for many machine learning applications such as the analysis of social networks, protein interactions, infectious disease, chemical reactions and molecules and these datasets contain privacy-sensitive data. This method provides a unique neural network design that can accommodate graphs with any structure.

Node classification and link prediction tasks have seen remarkable success with graph Neural networks. However, neither local nor global topological information is taken into account when processing the task of graph classification using a graph Neural network. As a result, developing graph neural networks to enhance the precision of graph classification has attracted increasing interest.

Here, we focus on the process of identifying the rich information embedded in a graph and extract valuable features for classification purposes. Although most existing GNNs are made for localized graphs with only local connections between nodes, graphs with more global connections but these methods show less accuracy when they are applied.

For this thesis we focus on GNNs model that can handle both localized and delocalized networks to get around this issue and reduce degree heterogeneity and degree homogeneity. Here, we incorporate different techniques by these GNNs to record both local and global information in the graph structure. As an illustration, whereas some GNNs employ message passing techniques to spread information throughout the graph, others use attention mechanisms to selectively attend to various regions of the graph but we will use these both mechanisms and create different types of datasets to experiment on the graph data.

Keywords - Artificial Neural Network, Convolutional Neural Network, Graph Neural Network, Graph Convolutional Neural Network, Graph Classification

Tables of Contents

1.Introduction

1.1 Motivation

The goal of developing a mechanism to predict a molecule's electronic structure based on its graph data is what drives the use of localized and delocalized networks with Graph Neural Network. Because a molecule's electrical structure controls its chemical properties, being able to precisely forecast it can be useful in a variety of applications. The Schrödinger equation must be solved traditionally in order to predict the electronic structure of a molecule, which is computationally expensive for big molecules.

The main concept is to visualize the molecule as a graph, where each atom is a node and the bonds between them are edges.to gather data from the graph at both the local and global levels. The localized network is concerned with the immediate surroundings of each node (atom), While the delocalized network takes into account the overall structure of the entire graph, the local network just takes into account the characteristics of the node and its neighbors.

We apply this method to use on a number of datasets of molecules with various sizes and complexity levels to show its efficiency. We developed a variety of datasets consisting of localized and delocalized networks and classified them using the Graph Neural network which has hubs in small portions of the graphs.

This dataset is tested on Binary class and multi class labels. so that in the real world our model can work on all types of classes. Compared to more conventional approaches, this method can accurately anticipate the electrical structure of these compounds.

1.2 Aim

The goal of this study is to develop a more effective and exact method for predicting the electronic structure of a molecule and reduce degree heterogeneity and degree homogeneity of localized and delocalized networks using GNNs. The technology could have a significant impact in a variety of scientific and engineering sectors by allowing for more precise and efficient synthesis of novel molecules and materials.

1.3 Challenges

There are several challenges associated with using localized and delocalized networks with Graph Neural Networks.

- Spectral data is frequently sparse and high-dimensional, making it challenging to extract useful features from it. This can make training accurate models that generalize well to new data difficult.
- In chemistry, labeled data is frequently scarce, making it difficult to successfully train models. Overfitting or subpar generalization performance may result from this.
- Chemical structures can have a wide variety of isomers and conformations, making them extremely complex. Because of this, creating precise models that can account for the variety of chemical structures can be difficult.
- The cost of computing to train Graph Neural Networks can be high, particularly for large datasets or complicated models. This may restrict these methods' ability to scale and make it difficult to effectively train models.
- It can be difficult to read the findings and comprehend how the model is producing predictions when using Graph Neural Networks because of their high degree of transparency. This may make it challenging to spot any potential biases or flaws in the models.

1.4 Method

In this method we remove from the graph spectral data, and then baseline adjustments and normalization are used because this phase is essential for getting the model to predict things correctly.

And we represent the spectral data and chemical structures as graphs by combining localized and delocalized networks such as erdos renyi graphs, Barabasi albert graphs and some other random graphs like star graphs, wheel graphs, cycle graphs, complete graphs and path graphs. While the delocalized network records the molecule's global connections, the localized network catches the molecule's local structure. Once combined together , these networks provide a single depiction of the molecule.

$$\widehat{A}\,(Normalized\ Adjacency\ Matrix) \;=\; D^{\frac{-1}{2}} \cdot A \cdot D^{\frac{-1}{2}}$$

Finally, we apply Graph Convolutional Network (GCN) which is the part of Graph Neural Network where we just normalize the adjacency matrix with selfloops and combine with the inverse square root of degree matrix and apply message passing step using GNN to learn features from the joint representation of the molecule.

The GCN is trained to predict the label and based on the label we assign to the adjacency matrix of the molecule, which is then converted into a molecular graph which has to be found as a real world application such as drug discovery, Contagion dynamics etc..

2.Background work

Basics of deep neural Network

2.1 Introduction:

Deep learning is a kind of machine learning that uses multiple-layered artificial neural networks to extract and learn high-level features from data. Deep learning aims to replicate how the human brain functions by creating intricate neural networks that can automatically learn from massive amounts of data.

Numerous applications, such as self-driving cars, medical image analysis, and recommendation systems, have found success using deep learning. Large datasets and powerful GPUs have made it feasible to train deep learning models with millions of parameters, giving them the potential to perform at the highest level on a variety of tasks.

There are two forms of machine learning approaches, supervised and unsupervised learning, that are used to train computers to generate predictions or find patterns in data.When training an algorithm with labeled data, where the input data and the associated output labels are known, supervised learning is used.

The algorithm picks up on patterns and connections between input features and output labels over time. When learning to predict outcomes, such as when classifying photographs or estimating the worth of a house based on its qualities, this form of learning is helpful.

Unsupervised learning, on the other hand, is teaching an algorithm on unlabeled data in which there are no output labels. The algorithm must independently find patterns and connections in the data.

This kind of learning is beneficial for grouping related data points together or spotting irregularities in a dataset.The objective of both supervised and unsupervised learning is

to build a model that generalizes well to fresh, unstudied data. Unsupervised learning can be helpful when labeled data is hard to come by or expensive to acquire, whereas supervised learning needs labeled data to train the model.

2.2 Least Curve Fitting

Least squares curve fitting is a method for determining the best-fit curve that describes the connection between two variables in a dataset. The purpose of this strategy is to discover a curve that minimizes the total of the squared discrepancies between the observed data points and the anticipated values from the curve.

The functional form of the curve that we want to fit to the data serves as the foundation for least squares curve fitting. Either a straightforward polynomial function or a more intricate function with several terms could be used for this.

2.2.1 Example :

Data−Points(x,y) :

$\{(1,3),(2,5),(3,7),(4,9),(5,11),(6,13),(7,15),(8,17),(9,19),(10,21)\}$

The equation of least square line is given by $Y = a + bX$

Normal equation for ‘a′: $\Sigma Y = na + b\Sigma X$

Normal equation for ‘b′: $\Sigma XY = a\Sigma X + b\Sigma X^2$

Solving these two normal equations we can get the required trend line equation
Thus, we can get the line of best fit with formula $Y = a + bX$

The final result of this method is a curve that best fits the observed data by reducing the overall difference between the actual data points and the expected values from the curve and it can be used in many disciplines, including engineering, physics, and finance, where it is crucial to describe and forecast the behavior of complex systems.

The difference between each observed data point and the matching predicted value from the curve is taken, squared, and added up over all the data points when using the least squares approach. The coefficient values are then changed to reduce this sum of squared discrepancies.

X	Y	X^2	XY
1	3	1	4
2	4	4	10
3	7	9	21
4	9	16	36
5	11	25	55
6	13	36	78
7	15	49	105
8	17	64	136
9	19	81	171
10	21	100	210
$\sum X = 55$	$\sum Y = 120$	$\sum X2 = 385$	$\sum XY = 825$

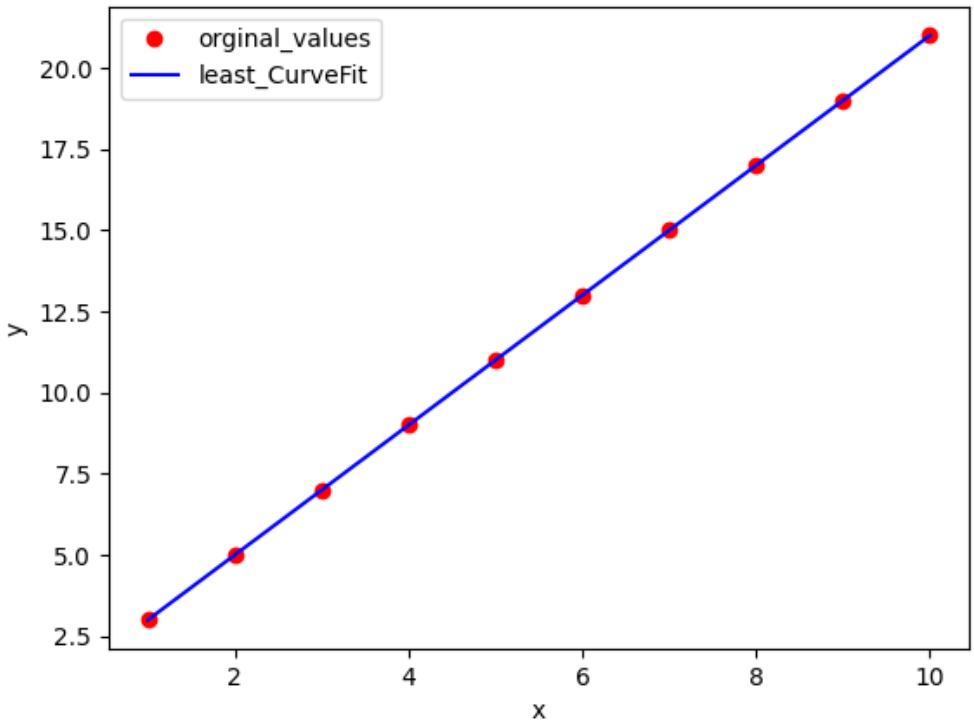
⇒ Normal equations are :

$$\sum Y = na + b\sum X \Longrightarrow 120 = 10a + 55b \Longrightarrow 2a + 11b = 24$$

$$\sum XY = a\sum X + b\sum X2 \Longrightarrow 825 = 55a + 385b \Longrightarrow a + 7b = 15$$

⇒ solving above equations we get $a = 1, b = 2$

⇒ least curve fit equation is $Y = a + bX \Longrightarrow Y = 2X + 1$



Above figure shows best Curve Fit

2.3 What is Simple Neural Network

A machine learning method known as a neural network is based on the structure and operation of the human brain. It is made up of a network of connected nodes, or neurons, that collaborate to process and analyze data.

The more complex the Network is, the more the layers are and neural networks with numerous layers of interconnected neurons are known as deep neural networks.Deep Neural Networks are able to learn from and extract high-level features from the input and are meant to handle and analyze huge, complicated datasets like photos, audio, and text.

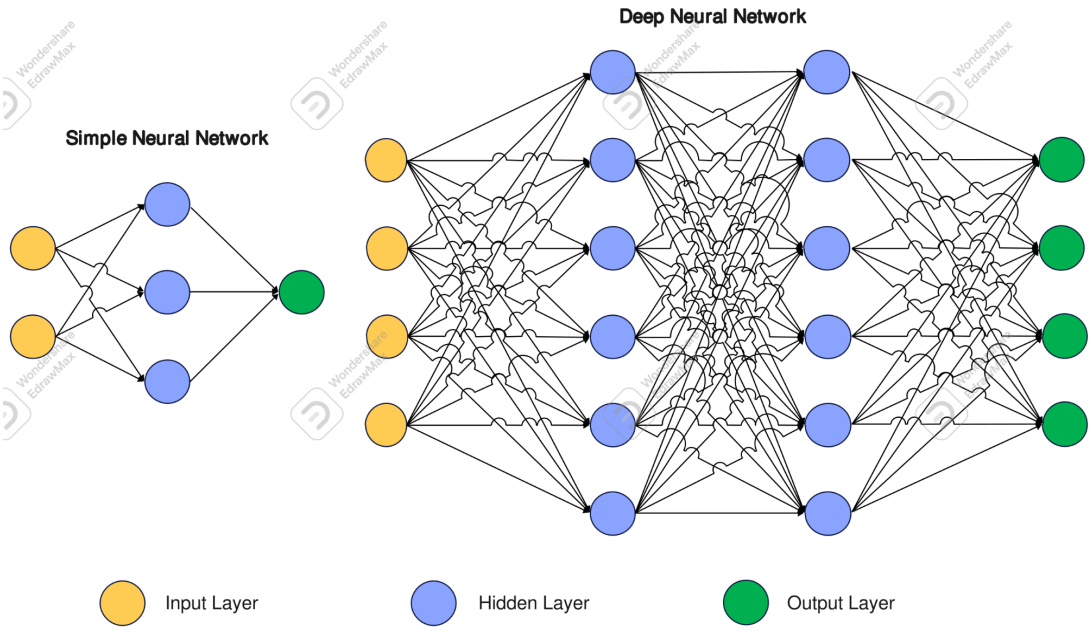
The common types of layers in Neural Network are Input layer, Hidden Layer, output Layer.

Input Layer : Data is entered into the input layer, which is the initial layer of the neural network. A feature or an input variable of the data being analyzed is represented by each neuron in the input layer.

Hidden Layer : Between the input and output layers of a neural network are levels known as "hidden layers." The input data is subjected to computations, and high-level features are extracted.

Output layer : The output layer, the last layer in a neural network, creates the network's output. The sort of task being carried out determines how many neurons are present in the output layer.

Visualization of simple and deep neural network



2.3.1 Operations in a single neuron ?

The simplest component of a neural network is a single neuron, often known as a perceptron. It processes one or more input signals before generating an output signal.

This can be done in a neural network's **forward** and **backward** propagations as part of the training process, in which the network learns to make predictions and adjusts

its weights and biases to minimize the error between its expected output and the actual output.

Forward Propagation

Here the process starts and moves from the input layer to the next layer so'on. Signals or input variables of the data being analyzed are supplied to the neuron as one or more input signals. Here each input signal is multiplied with weight and added to bias which determines the strength of the input signal's influence on the neuron's output.

$$Weighted\ sum\ =\ weights * input\ +\ bias$$

$$y\ =\ w \times\ x\ +\ b$$

The weighted sum + bias is passed via an activation function to produce the neuron's output. The activation function defines the neuron's firing rate and adds nonlinearity to the model.

$$Output\ =\ Activation(weighted\ sum)$$

$$output\ =\ \sigma(y)\ =\ \sigma(z)$$

We have different types of activation functions. The most commonly used activation function is sigmoid function, which generates an output between 0 and 1 and is employed in binary classification tasks and some Other activation functions are shown below

$$Linear\ Function \rightarrow Activation\ =\ weighted\ sum$$

$$Sigmoid\ Function \rightarrow Activation\ =\ sigmoid(weighted\ sum)$$

$$Sigmoid\ Function\ =\ \sigma(x)\ =\ \frac{1}{1+e^{-x}}\ \in\ (0,\ 1)$$

$$Tanh\ \ Function \rightarrow Activation\ =\ tanh(weighted\ sum)$$

$$tanh(x) \quad = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \in (-1, 1)$$

$$ReLU\ Function \rightarrow Activation \ = \ Max\{0, \ weighted\ sum\}$$

The neuron's output signal is the outcome of the activation function being applied to the weighted sum plus bias. It is the neuron's prediction or reaction to the input signals.

Backward propagation

Here the process starts and moves from the output layer to the previous layers and so'on.The output is compared to the original result by the model. To minimize error, weights are modified and the output is back-propagated. It goes through the process again and again to enhance accuracy and by evaluating the difference between the expected and actual outputs, the loss function is used to evaluate the network's performance.The most commonly used loss functions are shown below.

$$Mean\ squared\ error \ = \ \frac{1}{N}(\Sigma(\hat{y} \ - \ y))^{2} \quad N \in number\ of\ samples$$

$$Binary\ Cross\ Entropy \ = - \ [(y)log(\hat{y}) \ + \ (1 \ - \ y)log(1 \ - \hat{y})]$$

Here $\hat{y}$ is Predicted value and y is actual value

The Backward propagation objective is to reduce the discrepancy between the output that was anticipated and the actual output. Gradient descent is used to modify the neuron's weights and bias in order to achieve this. Calculating the gradient of the loss function with respect to the weights and bias, the weights and bias are then updated in the gradient's opposite direction.

The gradient of the loss function with respect to the weights and bias is determined

by propagating the error back through the network using the chain rule. The gradient of the loss function 'L' with respect to the neuron's output is first identified, and the chain rule is then used to calculate the gradient with respect to the weights and bias.

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \widehat{y}} \times \frac{\partial \widehat{y}}{\partial z} \times \frac{\partial z}{\partial w_i} \text{ and } \frac{\partial L}{\partial b} = \frac{\partial L}{\partial \widehat{y}} \times \frac{\partial \widehat{y}}{\partial z} \times \frac{\partial z}{\partial b}$$

Now we just have to find the values of $\frac{\partial L}{\partial \widehat{y}}$, $\frac{\partial \widehat{y}}{\partial z}$ and $\frac{\partial z}{\partial wi}$ by differentiating :

· $Here\ with\ respect\ to\ loss\ function\ we\ differentiating$

$$\frac{\partial L}{\partial \widehat{y}} = \frac{2}{n} \sum_{i=0}^{n} (\widehat{y}_i - y_i)$$

· $Here\ with\ respect\ to\ activation\ function$

$$\frac{\partial \widehat{y}}{\partial z} = \frac{\partial (z)}{\partial z} = 1, \ (Linear)$$

$$\frac{\partial \widehat{y}}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z)), \ (sigmoid)$$

· $Here\ with\ respect\ to\ weighted\ sum$

$$\frac{\partial z}{\partial wi} = \frac{\partial}{\partial w_i} (\sum_{i=1}^{n} (x_i \times w_i + b)) = \sum_{i=1}^{n} x_i$$

Now we do same for $\frac{\partial L}{\partial b}$ and find the values by differentiating The weights and bias are updated by subtracting the gradient of the loss function with respect to the weights and bias multiplied by a learning rate(α), which determines the step size of the weight and bias updates.

$$w_i' = w_i - \alpha(\frac{\partial L}{\partial w_i})$$

$$b' = w_i - \alpha(\frac{\partial L}{\partial b})$$

A hyper-parameter called learning rate(α) is used to regulate how much the bias and weights are altered.

2.3.2 Example

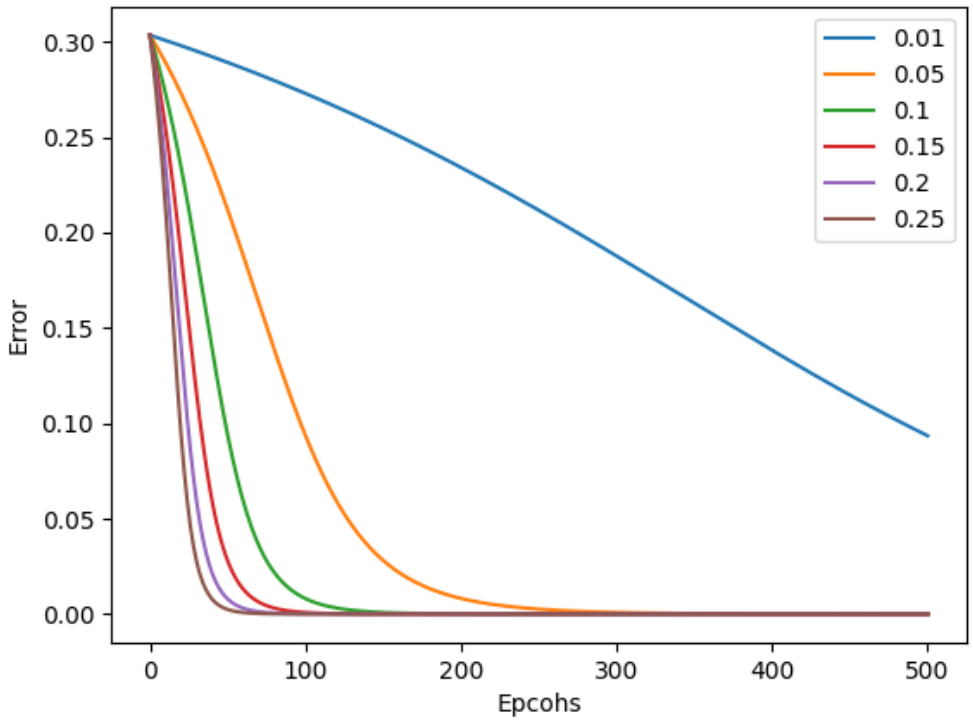
$input(x) = 1\ and\ output(y) = 0.33$

$weight(w) = 1\ and\ bias(b) = 1\ and\ learningrate(\alpha) = 0.1$

$ActivationFunction \ -->\ Sigmoid, i.e, \sigma(x) = \frac{1}{1+e^{-x}}$

Analysis:

- Below Figure shows that error is decreasing while increasing epochs and varying learning rates



2.4 Linear Regression Problem

A statistical technique for determining the relationship between a dependent variable (Y) and one or more independent variables (X) is called linear regression and By fitting a line to the data, linear regression seeks to reduce the difference between the dependent variable's expected and actual values. This line, which may be utilized to predict the values of incoming data points, illustrates the relationship between the independent and dependent variables.

We just gather both the independent and dependent variables (X and Y) data and we just Preprocess the data to allow us to deal with missing numbers, other problems and we Visualize the data to better understand the correlation between the independent and dependent variables.we Just solve this problem using a neural network.

2.4.1 Example

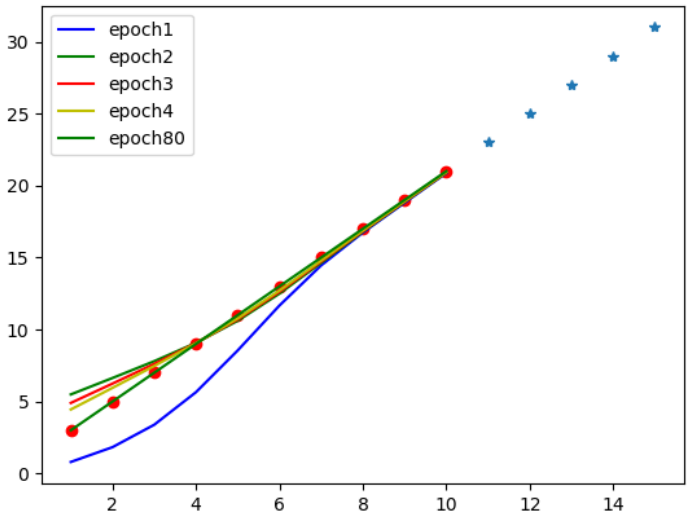
$Equation\ y = 2x + 1$

$Training\ Data\ for\ x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]\ and\ y = [3, 5, 7, 9, 11, 13, 15, 17, 19, 21]$

$Testing\ Data\ for\ x = [11, 12, 13, 14, 15]\ and\ y = [23, 25, 27, 29, 31]$

Analysis :

Here we see that model is trained used trained data got the best fit line and tested for the given data points (*).



2.5 logistic Regression Problem

Logistic regression is a statistical method used for binary classification problems,

where the dependent variable is binary (0 or 1).Fitting a logistic curve to the data in logistic regression entails determining how the probability of the dependent variable being 1 as a function of the independent variables will look on the curve. By calculating the likelihood that the dependent variable will be 1, given the values of the independent variables, this curve can be used to make predictions for new data points.

Here we can not use logistic regression because Linear regression is used for continuous outcome variables based on one or more input variables.While linear regression assumes a linear relationship between the input factors and the result variable, logistic regression can capture nonlinear relationships between these variables. Logistic regression may offer a better fit to the data if there is a non-linear relationship between the input factors and the result variable.

2.5.1 Example

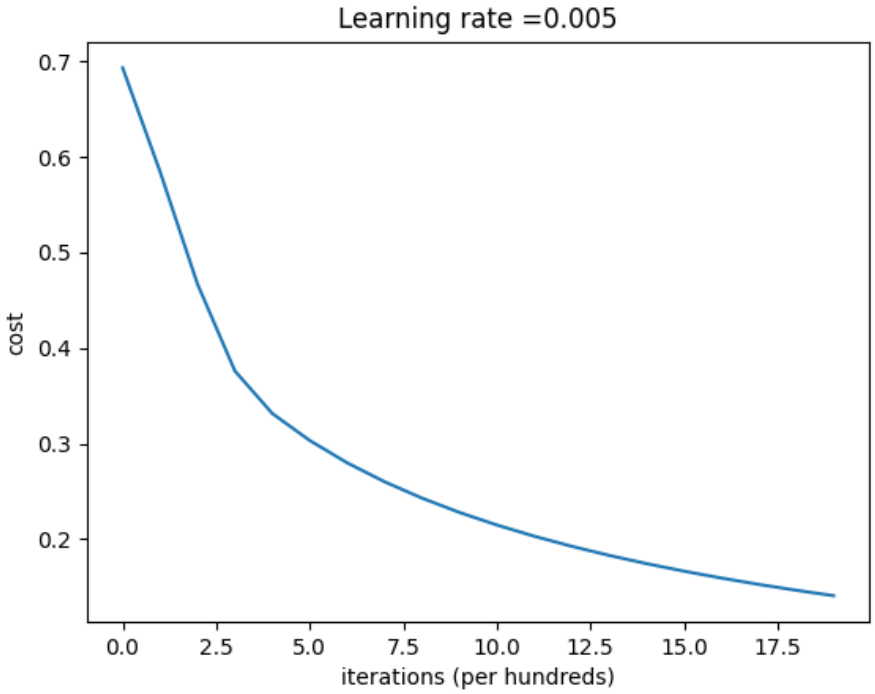
Here we take an Example on logistic regression problem for predicting whether an image contains a cat or a dog.

First, we compile a dataset of pictures of dogs and cats.we make the pixel values more manageable for the model, we preprocess the photographs to make sure they are of the same size and format. and we label the pictures as "cat" or "dog" to make a binary classification issue.

We take these input pixels and input to the Neural network, here we use Binary cross entropy loss function and sigmoid activation function to predict and calculate the loss.

Finally the model will learn from the given images and we get the best non-linear curve so that we test on some images and it will predict either ‘Dog’ or ‘Cat’.

Below figure shows Error Vs Loss for logistic regression model



Here we used small images but in real life there can be large images but it is not possible to input that much huge amount of data into the neural network because it may lead to decrease in accuracy and can not predict correctly.

When working with picture or video data, which have a two-dimensional structure, neural networks treat the input data as a one-dimensional vector, which can be constricting. Traditional neural networks also need a lot of parameters, which increases their processing cost and makes it challenging to train them on large datasets.

CNNs were developed to get over these restrictions and perform better on image and video recognition tasks. Convolution is a method used by CNNs that enables the network to learn spatial elements like edges, lines, and textures from the input image without requiring a significant number of parameters.

This we will see in the next work……

Convolutional Neural Network

2.6 CNN Introduction

Deep learning neural networks known as convolutional networks (CNNs) are created specifically for processing and analyzing image and video input. The human visual cortex, which is in charge of processing visual input in the brain, served as an inspiration for the design and operation of CNNs.

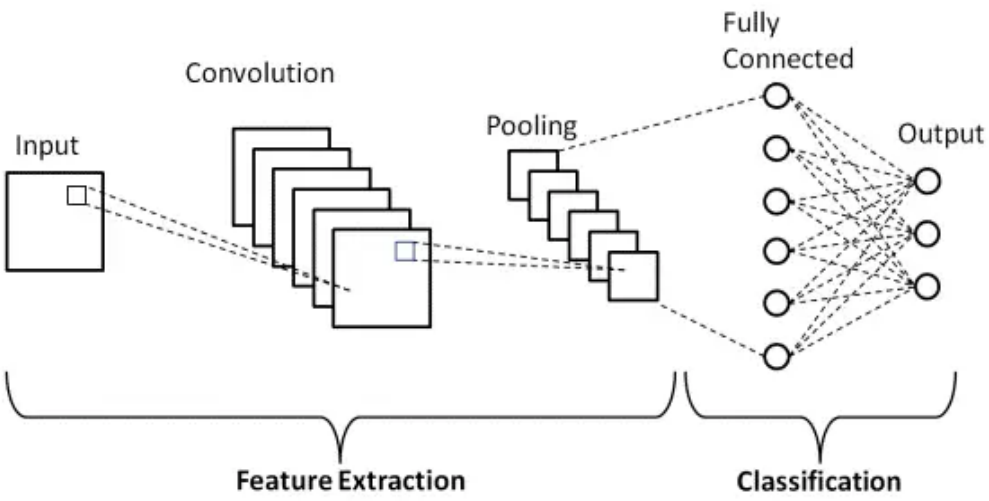
CNNs use a technique known as convolution to extract spatial information from the input image, such as edges, lines, and textures. During the convolution operation, a tiny filter or kernel is slid over the input image, and the dot product between the filter and the corresponding pixels in the image is computed. This process is repeated for each location in the input image once, producing a feature map that illustrates the distribution of different spatial features across the image.

Convolutional, pooling, and fully linked layers are only a few of the many layers that make up a CNN. The class probabilities for the input image are generated using the output of the last layer after it has been passed through a softmax function.

Many different image and video identification tasks, including object detection, picture segmentation, and facial recognition, have seen great success with CNNs. In other fields where the input data has a structure akin to that of photos and videos, such as natural language processing and speech recognition, they have also been used.

CNNs can also learn to identify these features at different scales or resolutions, which is important for recognizing objects of different sizes in an image.

2.7 CNN Architecture



Implementation : There are 5 main layers in CNN Architecture:

1) Input Layer : The input layer, which is the initial layer of the CNN, applies a series of filters or convolutions to the input image. Edges, corners, and textures are just a few of the qualities that these filters may identify in an image. The input image's size, which is often displayed as a three-dimensional matrix of width, height, and depth, determines the size of the input layer.

2) Convolutional layer : The center of the CNN, where the actual processing happens, is the convolutional layer. The input image is subjected to a set of filters or kernels(Box line filter, vertical line filter, Diagonal line filter), which convolve or slide over the image and perform a dot product between the filter and the input's local receptive field. This process creates a feature map that draws attention to specific features in the image. The filter, stride, and padding sizes all affect how big the feature map will be.

3) The activation layer introduces non-linearity and enables the network to learn complicated patterns and relationships in the data by applying a non-linear activation function to the output of the convolutional layer. Here we are using ReLU which maintains the positive values while setting all negative values to zero.

4) Pooling Layer : By downsampling the feature maps, the pooling layer makes them smaller. The most popular kind of pooling is called max pooling, which chooses the maximum value from a set of values and discards the other values. The feature maps' dimensionality is decreased through pooling, which also increases the network's computational effectiveness.

5) Fully connected layer(From Here it is the same as Simple basic Neural Network) : Conventional neural network layer, computes the network's final output using the preceding layer's flattened output. Backpropagation is used to train a set of learnable weights and biases in the model. The fully connected layer's output is determined by the particular function that the network is intended to carry out.

   Here Input Layer, Convolution Layer and Pooling Layer is used for important feature extraction and Fully Connected Layer and Output Layer use for classification.

   Here we saw that CNNs are a powerful tool for image recognition and other visual tasks, but there are still some challenges and limitations associated with them.

   1) Over fitting
   2) Computationally Expensive
   3) CNNs may miss important context in the image while extracting

   The main problem in CNNs is if the size of the input image varies then they can not extract information because CNNs work on dimensions and grids which are very difficult to Handle so that GNNs are introduced.

   This we will see in related work ……

3.Related Work

Graph Neural Network

3.1 Introduction:

Graphs can be used to represent most real-world data. Objects can be denoted as nodes of the graph and edges can be used to represent relationships between them.Graphs are used in almost every field.

In social networks, graphs are used to provide online recommendations, implement news feeds and calculate page rank. In the field of neuroscience, neurons are denoted by nodes and connections between them as edges.

These graphs are then used to analyze the functionality of brain networks.GNNs are being used in real-world scenarios like Social Networks, Drug Discovery, Fraud Detection, Protein Folding, Contagion dynamics etc…

There are many more applications in other fields. These prove the importance of working with graphs.Graph neural networks (GNNs) are a powerful class of deep learning models that are specifically designed for graph-structured data and they have been shown to be highly effective in capturing the structural information and dependencies present in graphs.

GNNs have proven to be highly valuable and suitable for a wide range of real-world applications due to their ability to model and analyze graph-structured data, such as node classification, graph classification, graph generation, and link prediction, among others.

GNNs can capture both local and global graph information, and adapt to varying graph structures and sizes and they can handle graphs with varying number of nodes, edges, and node features, making them suitable for graphs with different levels of complexity and heterogeneity.

3.2 Importance of graph classification

Graph classification tasks are of significant importance in various real-world applications due to their ability to capture and analyze complex relationships and dependencies among entities represented as graphs.

- Predicting Graph Properties
- Understanding Graph Data
- Enabling Graph-Based Machine Learning
- Handling Heterogeneous Graphs
- Scalability
- Real-World Applications

Accurate graph classification can lead to improved decision-making, better resource allocation, and enhanced performance in various real-world scenarios.

3.3 Why GNNs are used over ANNs

Graph neural networks are specifically designed to operate on graph-structured data, which is different from the tabular or sequential data which are designed for traditional neural networks.

When dealing with non-Euclidean material that is represented as graphs or networks, Graph Neural Networks are preferred over Artificial Neural Networks. The handling of graph-structured data is outside the scope of ANNs, despite the fact that they are quite good at handling structured data like vectors and matrices.

GNNs are used over traditional ANNs for graph-structured data because they are capable of capturing the complex relationships and structures in graphs, learning both node-level and graph-level representations, handling graph variability, scalability, incorporating domain-specific features, and providing interpretable results. These capabilities make GNNs highly suitable for a wide range of real-world applications that involve graph-structured data.

The structure information of graphs can be captured by GNNs because they can learn representations of nodes and edges that take connection information into account. GNNs may handle graphs of various sizes and shapes. This increases the flexibility of GNNs when dealing with data that comes in different sizes and shapes, including chemical structures.

3.4 Why GNNs are used over CNNs

When dealing with non-Euclidean data that is represented as graphs or networks, Graph Neural Networks are preferred over Convolutional Neural Networks . Graphs and other irregularly structured data cannot be handled by CNNs, despite the fact that they excel at extracting features from typical grid-like data like pictures.GNNs, on the other hand, are created to function on graph-structured data, making them suited for tasks like graph classification, node classification, and link prediction.

GNNs may learn representations of nodes and edges that take into account the connectivity information between them, allowing them to capture the structural information of graphs.

Additionally, unlike CNNs, which require an input of a specific size, GNNs can handle graphs of various sizes and shapes. This increases the flexibility of GNNs when dealing with data of various sizes and types, such as social networks or chemical structures.

3.5 Implementation

In the Graph classification problem, given is a family of graphs and a group of different categories, and we aim to classify all the graphs into the given categories and predict some output. Here we just Load and preprocess the graph data by reading graph data from a file and converting it into a suitable format that can be fed into a GNN.

The graph data should typically include information about nodes, edges,adjacency matrices and their attributes because the graph consists of G = (V, E) Nodes(V), Edges(E) and attributes.

3.5.1 Forward Propagation

1) We initialize every node in the graph with feature vectors which are node attributes and are represented by feature vectors. Here Each node sends a message to its neighboring nodes. The message sent by node i to node j is a function of the feature vectors of node i and edge i-j. This is called message passing.

2) Aggregation Step : Each node aggregates the messages received from its neighboring nodes. Here We just take the (A)Adjacency matrix(5x5) and (X)Feature matrix(5x2) (feature vector for each node is tilled into a matrix of size(Number of Nodes)) we take these two matrices and do dot product store in B(5x2)(output matrix).

$$\Rightarrow B_{5\times2} = A_{5\times5} \cdot X_{5\times2}$$

3) Neighborhood Aggregation Step : Here we just take the output Matrix(B)(5x2) from the aggregation step and weight matrix(2x2)(here weight matrix the values are taken randomly and the size of the matrix is decided based on the size of feature vector dimension) we take these two matrices and do dot product by adjusting the dimension of matrix 'B' and store in X(5x2).

$$\Rightarrow X(updated)_{5\times2} = B_{5\times5} \cdot W_{5\times2}$$

By observing the B matrix we understand that those values that are present in the matrix are the sum of node degrees for the current node and its neighborhood where each node aggregates the information from its neighboring nodes. If we continue these two steps it will cover the whole graph and aggregate the information from its neighboring nodes. we get the updated feature vectors for each node.

4) Relu : The ReLU function is applied element-wise to the (Updated feature vector)X of the message passing or node update step in the GNN, where each node's aggregated information from its neighbors is combined with its own features.

$$\Rightarrow C(relu)_{5\times2} = C_{5\times2}$$

Relu function allows it to learn complex patterns and representations from the graph data. It ensures that only positive values are passed forward, while values less than zero are set to zero. This helps in capturing important features and discarding irrelevant or negative information from the graph data.

5) Readout : In the Graph level representation Readout function will sum up the node representations(feature matrix 'X') across the graph element-wise to generate a fixed-size graph-level representation.

$$\Rightarrow C(readout)_{5\times2} = C_{1\times2}$$

This captures the global information of the graph. This graph-level representation is then used as input to a classifier to make the final prediction for the graph classification task.

6) Linear function : It is the same as a simple basic Neural network.

$$S = A(weight) \cdot C(output\ of\ readout\ function) + b(bias)$$

$$Sigmoid\ Function = \sigma(S) = \frac{1}{1+e^{-S}} \in (0, 1)$$

7) Binary classification : Here we just input the sigmoid output(P) value to get the predicted class between either 0 or 1. This can be done by applying some conditions.

$$if\ P > 0.5\ then\ 1\ and\ P < 0.5\ then\ 0.$$

3.5.2 Loss Function

Here we use the Binary cross entropy loss function which is commonly used for binary class graph classification problems where the goal is to predict binary outcome.It is defined as y(label), z(linear function).

$$Loss = y(log(1 + e^{(-s)})) + (1 - y)(log(1 + e^{(s)}))$$

Now we check if the loss is zero or close to zero. If not then we have to decrease the loss by updating the parameters ‘W’, ‘A’, ‘b’ and this is done by backpropagation.

3.5.3 Backward propagation

After the forward propagation is finished, the gradients of the loss with respect to the GNN's output can be calculated using conventional back propagation methods. An optimisation approach is used to compute the gradients of the loss with respect to the model's parameters and update them in the backward propagation procedure for GNNs.

To calculate the gradient with respect to the model parameters, the gradients computed in the preceding step are back propagated through the GNN. This entails computing the gradients with respect to the model parameters by chaining together the partial derivatives of the message passing, aggregation, and update functions with respect to their inputs and weights.

The computed gradients are used to update the parameters of the model using an optimization algorithm such as stochastic gradient descent or Adam which is mainly used for graph classification tasks.

These steps are repeated until the parameters of the GNN converge to their optimal values.

4. Proposed method

4.1 Problem Statement

In this research, we develop a method to classify the localized and delocalized networks based on Graph Neural Networks (GNNs), to efficiently and accurately recreate chemical structures from spectral data.

The capacity to predict the properties of molecules can have a substantial impact on the design of new medicines and materials, which makes this problem particularly important in the fields of materials research and drug discovery. The suggested approach seeks to get over issues with this work such data sparsity, a lack of labeled data, complexity of chemical structures, and computing complexity. By creating a precise and effective technique for deriving chemical structures from spectral data.

4.2 Architecture

Graph convolution is the fundamental principle that drives most GNN systems. We essentially try to apply the concept of convolution to graphs. Graphs can be thought of as a generalization of photographs, with each node representing a pixel connected to eight (or four) other pixels on either side.

Here we will be focusing on Graph classification using GNN.

1) We enforce self-connections by adding the identity matrix I to the adjacency matrix A to get the self looped adjacency matrix $\widehat{A}$.

$$\widehat{A} = A + I$$

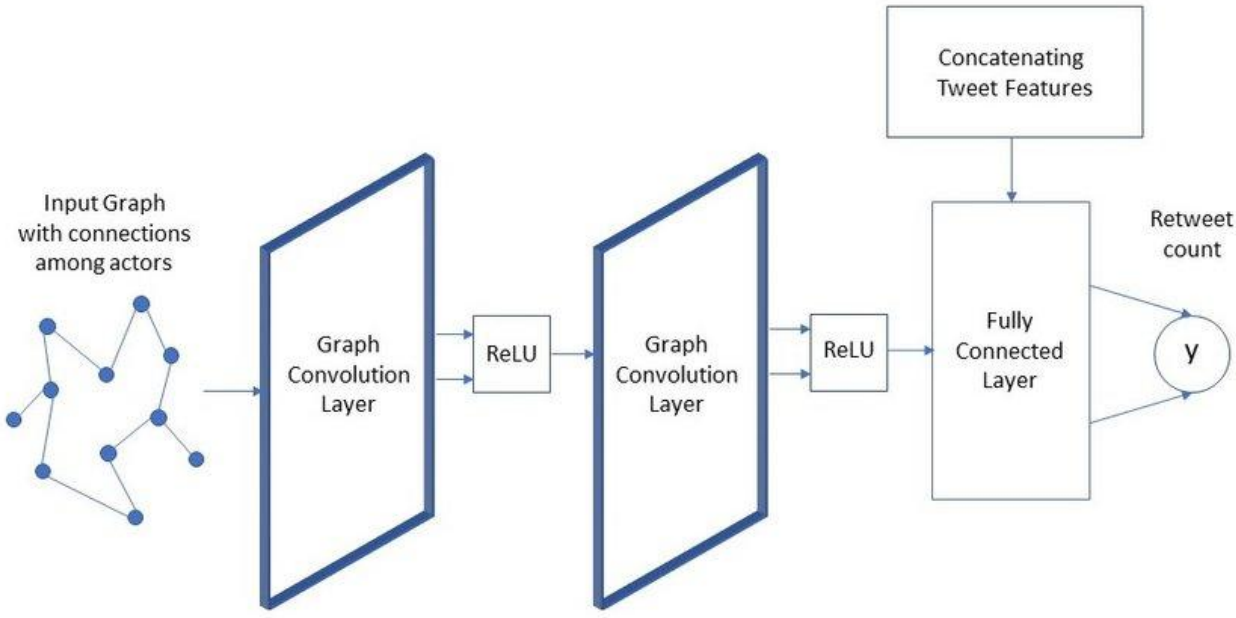
2) We do symmetric normalization by inverse square degree matrix.

$$\widehat{A}\,(Normalized\ Adjacency\ Matrix) = D^{\frac{-1}{2}} \cdot \widehat{A} \cdot D^{\frac{-1}{2}}$$

3) We just take the Adjacency matrix($\widehat{A}$) and Feature matrix(H) (feature vector for each node is tilled into a matrix of size(Number of Nodes)) we take these two matrices and do dot product and we output matrix of this operation is again multiplied with weight matrix(W) and then we get updated feature vectors for each node in the graph.

$$H = \sigma(D^{\frac{-1}{2}} \cdot A \cdot D^{\frac{-1}{2}} \times H \times W)$$

4) These three steps are operations in the convolutional layer and we apply Relu activation function and pass it to Linear part simple basic neural Network. Here if we increase the convolutional layers, each node will learn more features from neighboring nodes and aggregate themselves and finally the model will learn more about the graph and predictions will be accurate.



4.3 Localized and Delocalized Networks

Localized Networks : A particular form of neural network architecture called a localized network is made to identify regional patterns and connections in data. A localized network in computer vision might be made up of several convolutional layers that act on discrete areas of an image, allowing the network to recognise local elements like borders and corners.

Delocalized Networks : A delocalized network is one that lacks considerable local structure or clustering and nodes that are tightly packed together. In contrast, the network exhibits a more uniform distribution of connections among the nodes, with connections spread out fairly equally in the network

Here we applied GNN architecture on localized and delocalized Networks.Firstly we trained 400 networks(localized delocalized networks) and 100 for testing with 3 GCN Layes and accuracy not so good but we generated 5000 networks and trained 4000 networks and tested 1000 networks and predictions are accurate.

After doing a lot of research and experiments, we will notice a large difference between the highest and lowest degrees in the network and as nodes with high degrees may dominate the network and skew the learning process. and we observe a more even distribution of node degrees in the network this may lead to limited variability in the training data or difficulty in distinguishing between similar nodes.

Our method used to balance the influence of high-degree and low-degree nodes in the network is known as symmetric normalization. So that the sum of the squared message weights across all nodes is equal to one, symmetric normalization normalizes each node's message by the square root of its degree. This makes high-degree nodes less significant and guarantees that each node's significance correlates with its degree.

We understand that degree heterogeneity and degree homogeneity is very problematic for GNNs and this can be solved by linear dynamics.

5. Experiment

5.1 Methodology:

Graph neural networks are a class of neural networks that can operate on different types of Networks and are particularly useful for graph classification problems. Here we used TUD datasets which are extracted from the same background. So to check the performance and to analyze the relation among different types of graphs like random networks, scale free networks, star graphs, complete graphs, cycle graphs, path graphs and wheel graphs which are of different backgrounds.

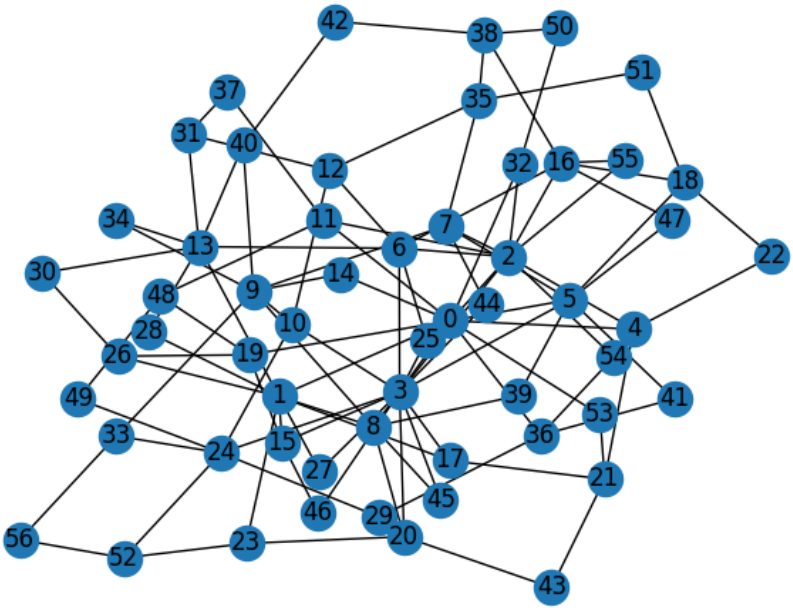
Here real world problems are of different backgrounds which have different graph properties.However real world graphs are a bit complex and noisy. Here we have generated a dataset (Collection of these localized and delocalized Networks) for graph classification.

5.2 Data collection:

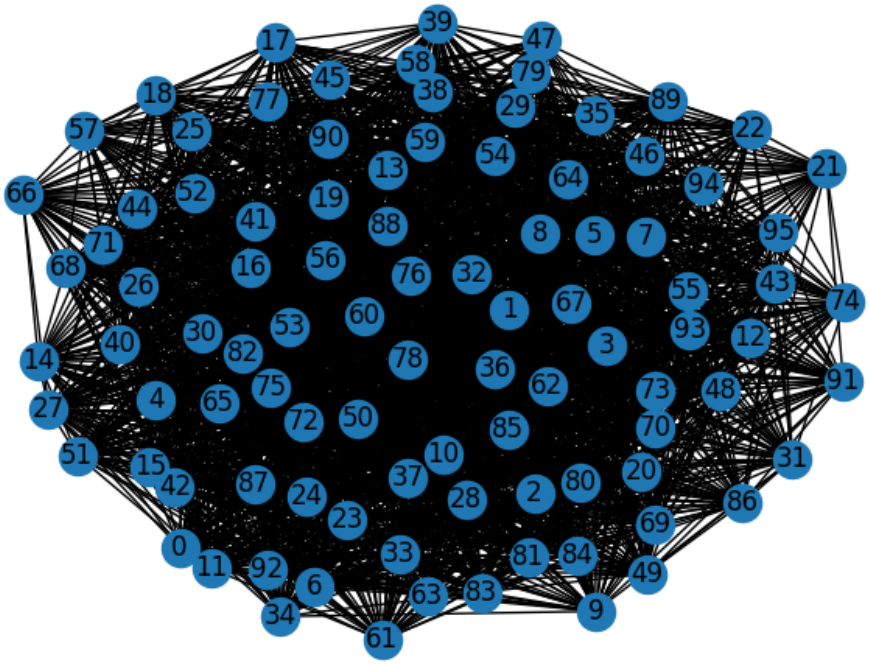
Graph neural networks are a class of neural networks, We generated some barabasi albert graphs of ``n`` nodes(generated randomly between 30 to 100, can be varied in future) is grown by attaching new nodes each with ``m`` Edges(chosen randomly between 2 and 3, can be varied in future) that are preferentially attached to existing nodes with high degree and extracted adjacency matrix for each graph and created edge_indices, generated feature matrices for each graph based size of the graph and assign labels.

We generate the same for Erdos renyi graphs(The G(n,p) model chooses each of the possible edges with probability (p(0.5 to 0.8), can be varied in future)) and other random graphs and create edge_indices, feature matrices and assign labels.we generated these graphs based on average node degrees.

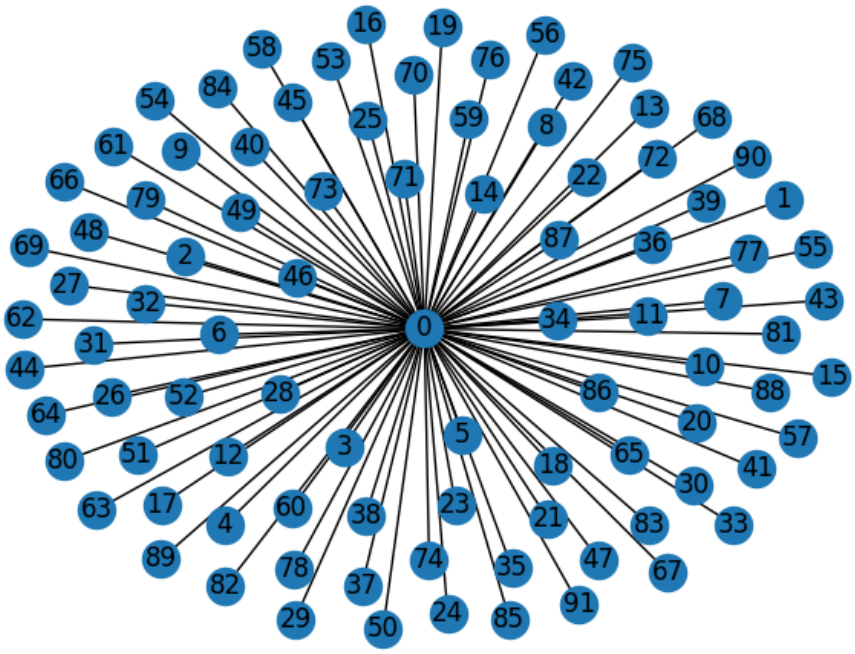
- Barabasi albert graphs : Label is `` **1** ``



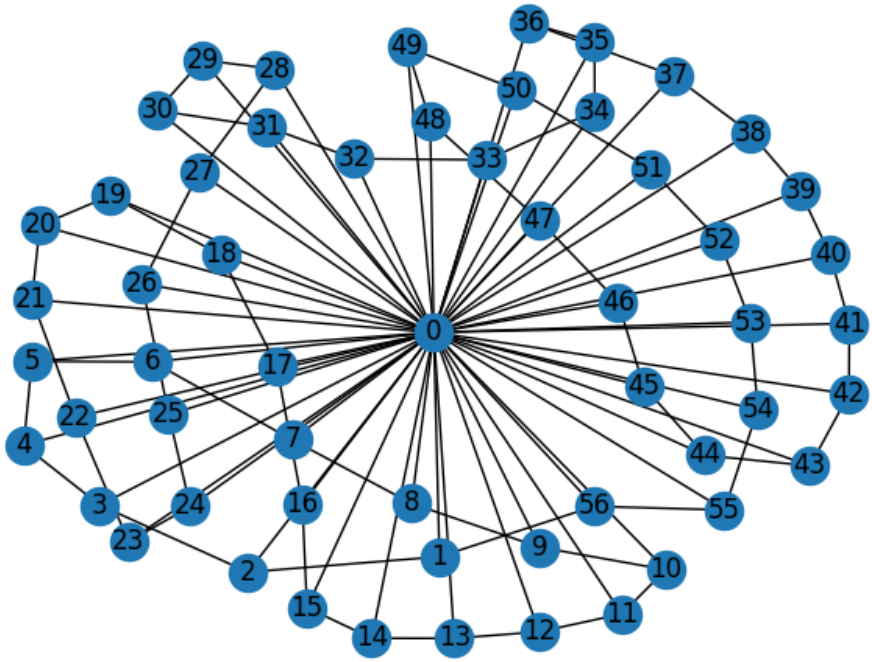
- Erdos renyi graphs : Label is `` **0** ``



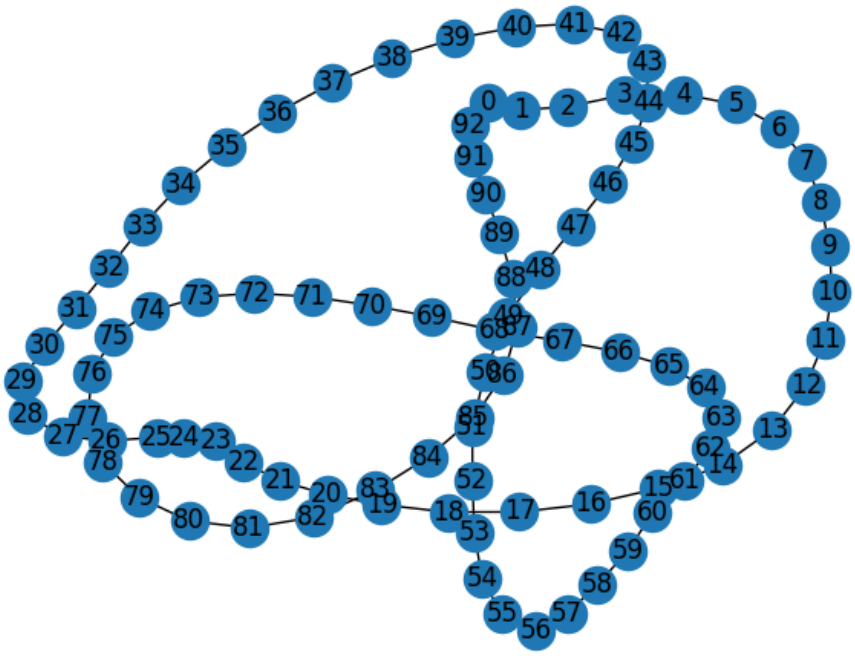
- Star graphs :Label is `` **1** ``



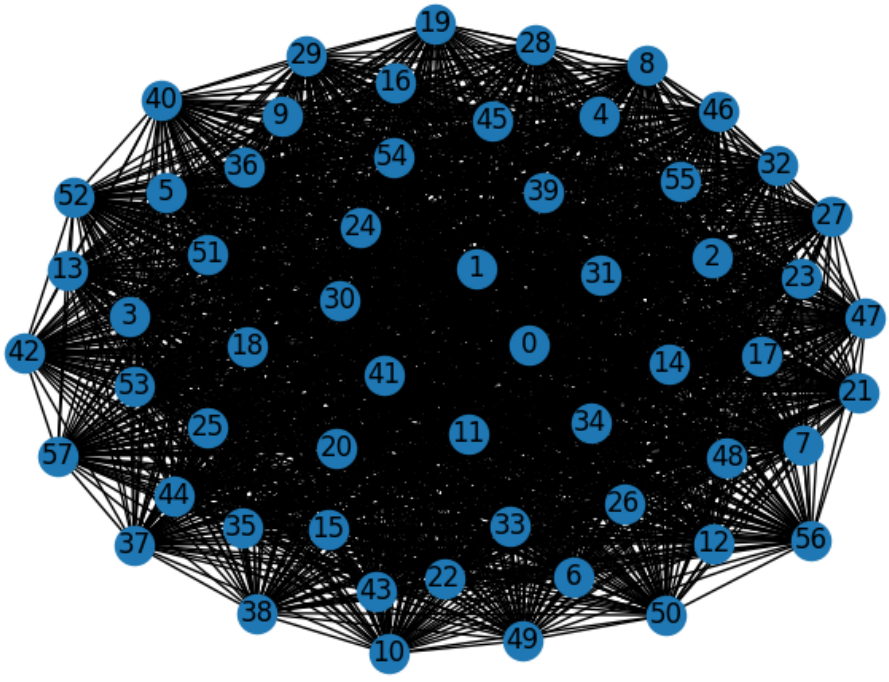
- Wheel graphs :Label is`` **1** ``



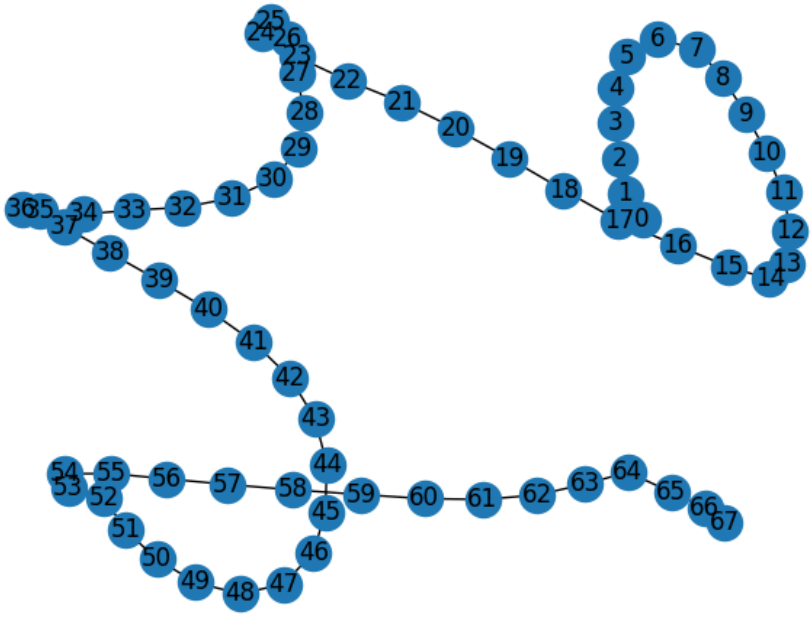
- Cycle graphs : Label is `` **0** ``



- Complete graphs : Label is `` **0** ``



- Path graphs : Label is `` **1** ``



5.3 Data Preprocessing:

The graph data we have extracted is processed using torch geometric data which is imported from the torch geometric python library consisting of Data class and Dataset class where our graph data is converted to torch geometric class so that the processed dataset can be used for training and testing sets.

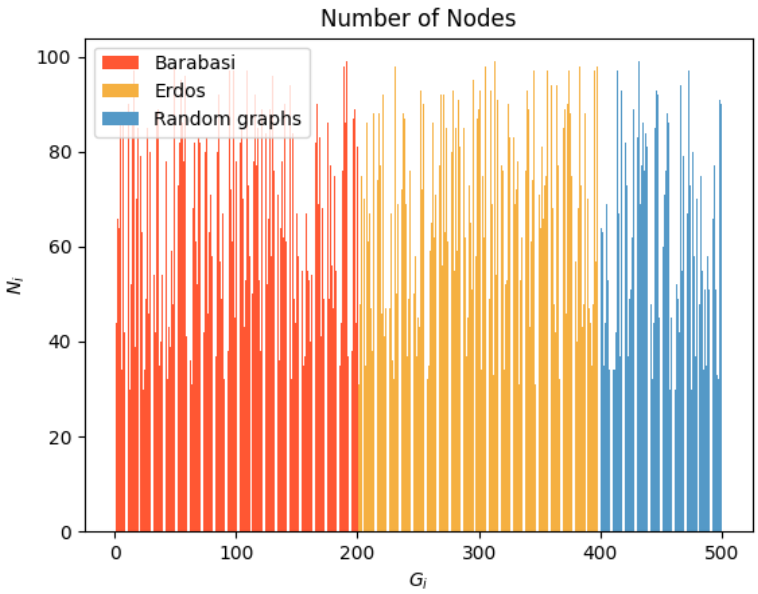
Steps for Data preprocessing **:**

- **Data Loading :** Edge-indices, feature matrices and labels are passed into the **Data** object which is provided by torch geometric library.
- **Cleaning the Data** : The data may contain missing values, outliers, or other inconsistencies. Cleaning the data involves identifying and handling these issues.
- **Feature Normalization**: The features of the nodes in the graph may have different scales, which can negatively impact model performance. Torch Geometric provides functions to perform normalization

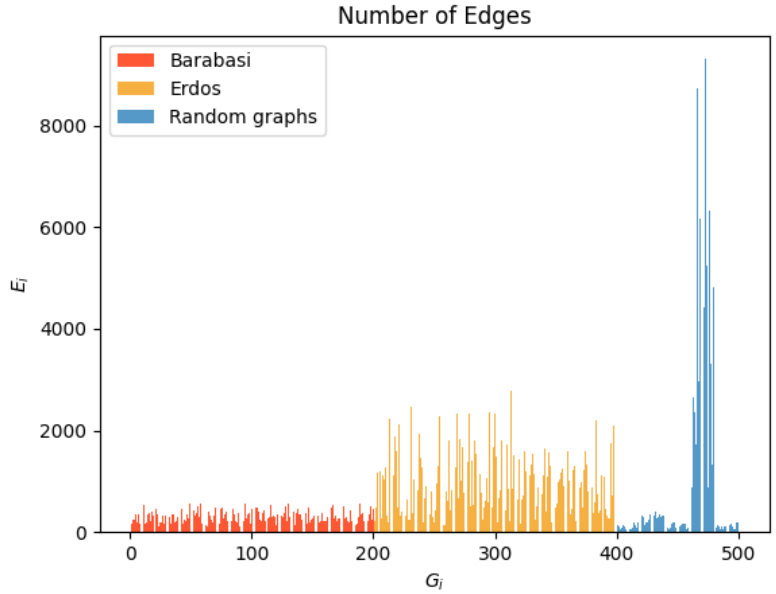
- **Graph Transformation**: Here some graph models in Torch Geometric require the graph to be transformed into a specific format.Torch Geometric provides functions to perform these transformations.
- **Feature Extraction**: Here graph features may not be in a suitable format for the model. Feature extraction is the process of converting the graph features into a different format that is more compatible with the model. Torch Geometric provides various functions to perform feature extraction.
- Finally **Data** objects are stored in the List (Dataset) and This dataset is shuffled splitted into training set and testing set and these two are passed into **DataLoader** class which is imported from the torch geometric library.
- DataLoader loads a dataset and divides training data and training data into batches and these batches are passed one by one to the model.
- Finally we can perform **graph classification** on these processed data very easily.

5.4 Analysis:

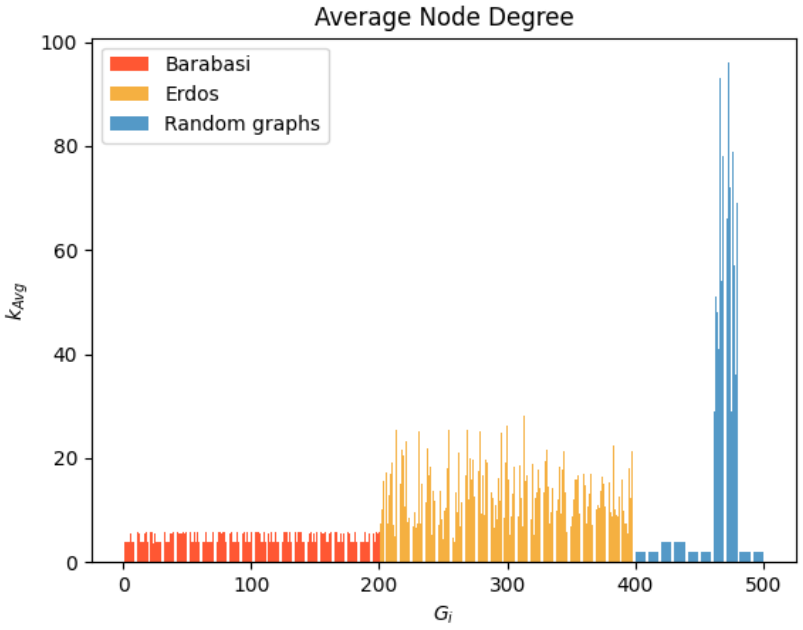
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



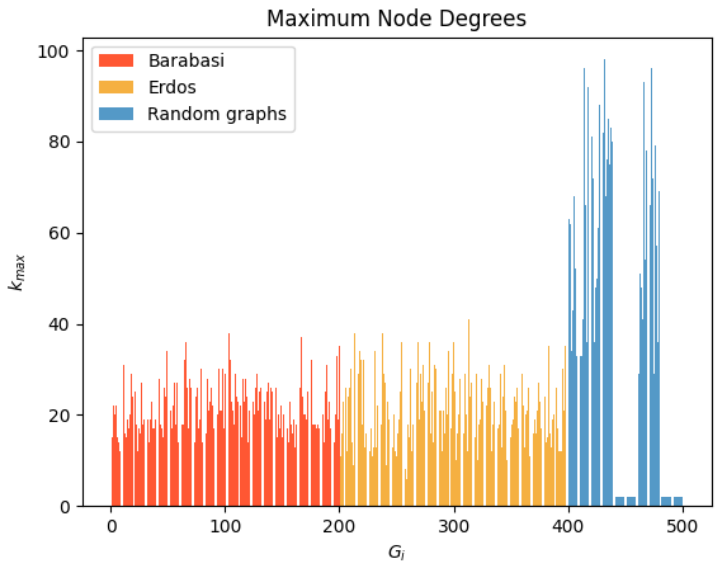
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



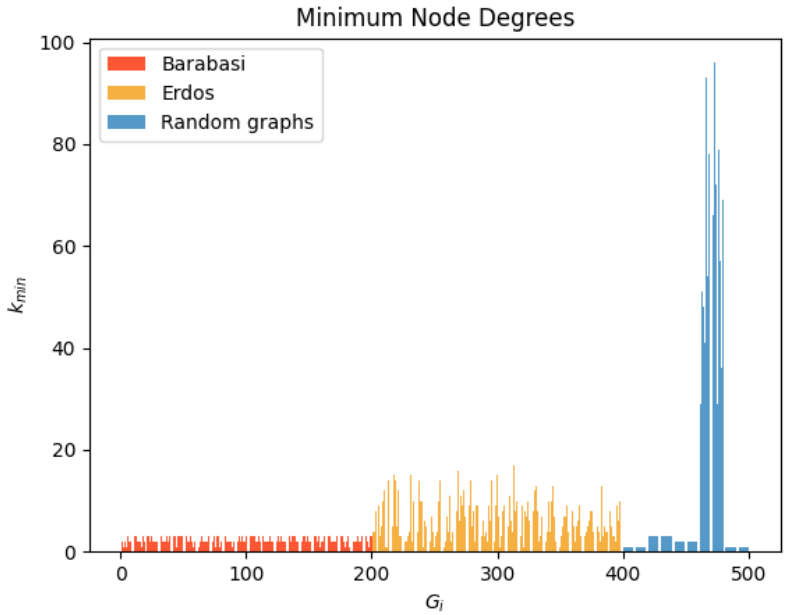
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



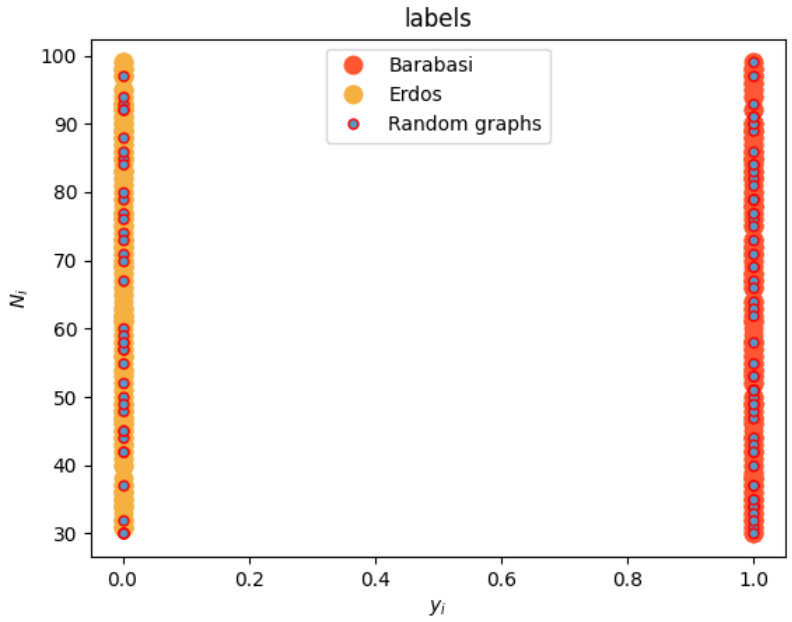
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



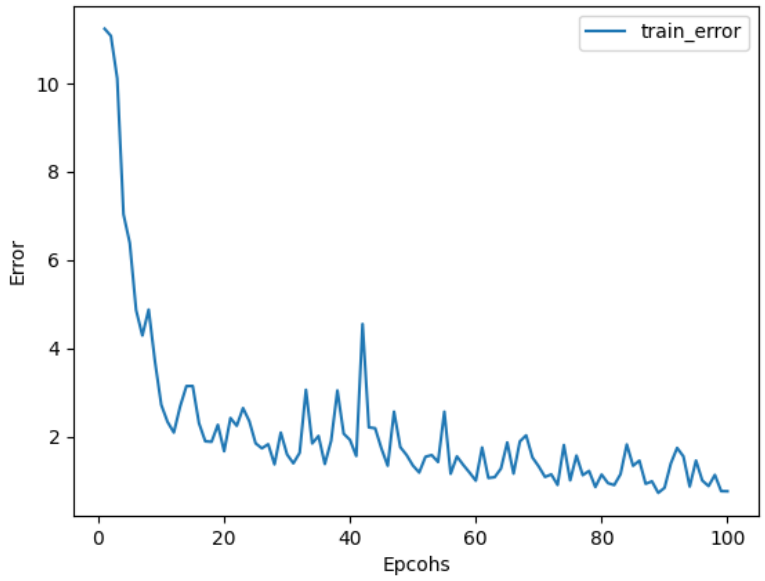
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



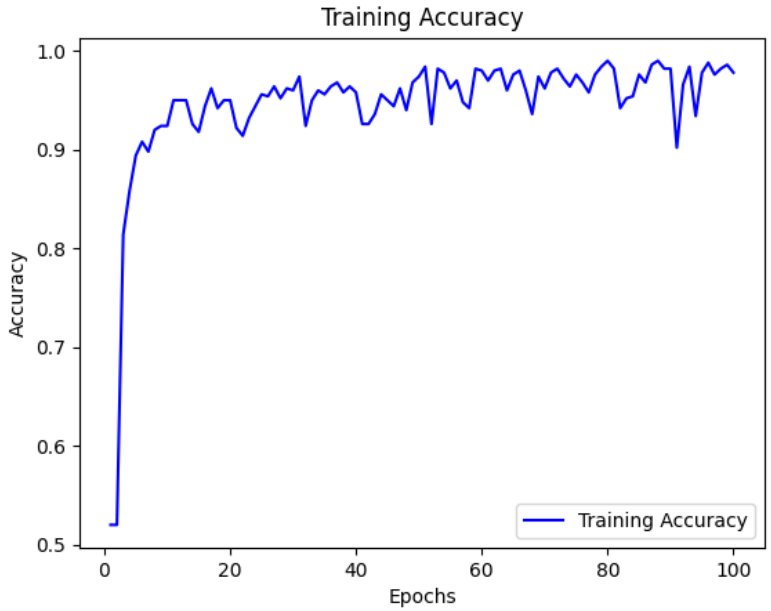
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



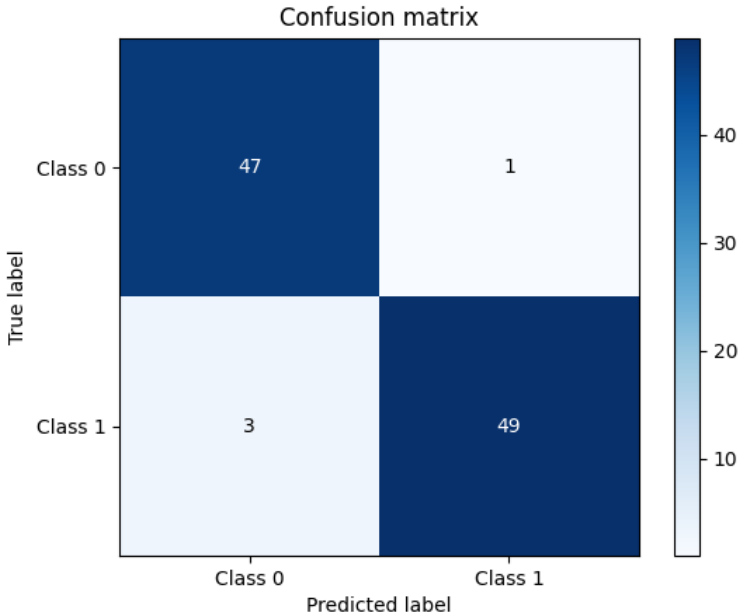
- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



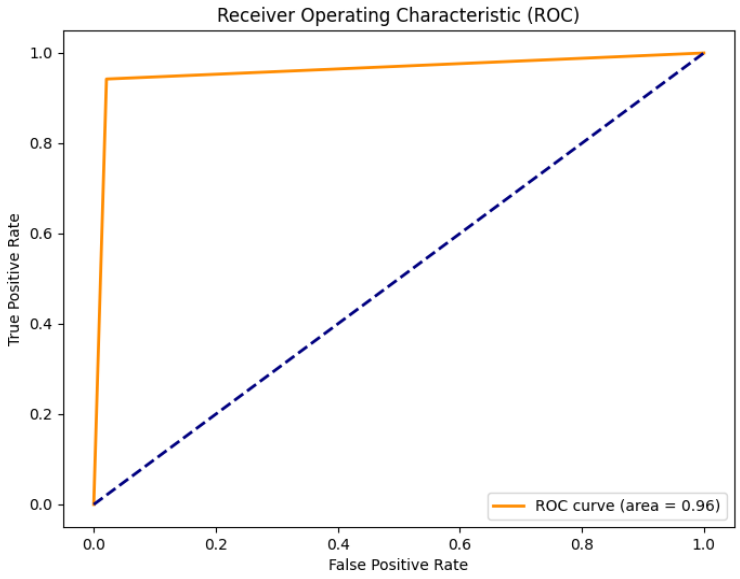
- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



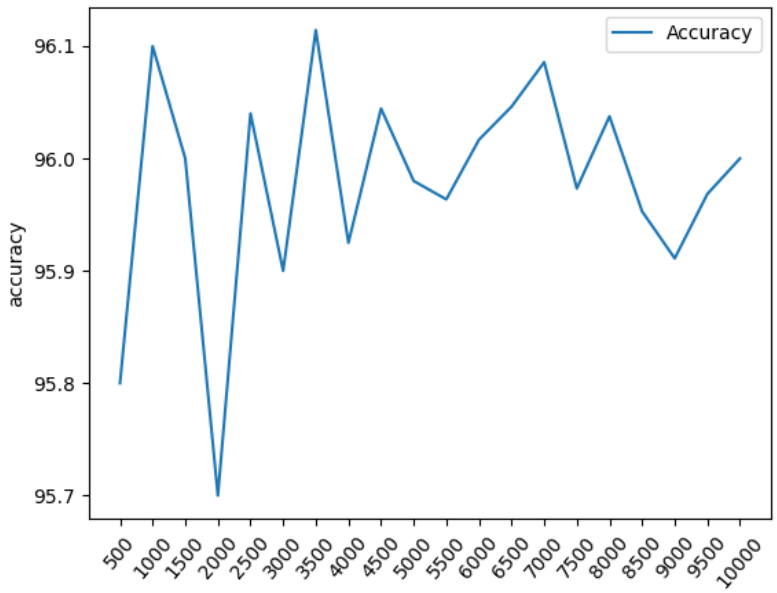
- Plotting a confusion matrix for the dataset tells us how many 0 class labels are positive and 1 class labels are positive.



- Plotting a ROC curve for the dataset shows us the major difference between the true positive rate and false positive rate of the model



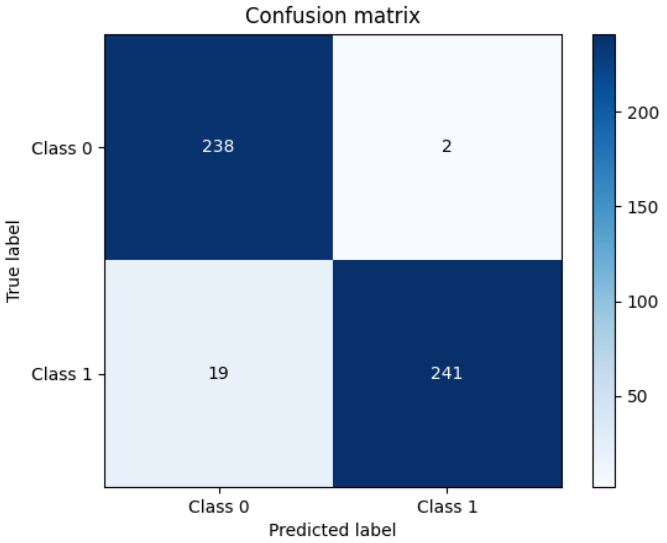
- Below plot shows the accuracy of different sizes of dataset used for the experiment.By this we understand that difference is not so big and maintain the same accuracy for large datasets also.

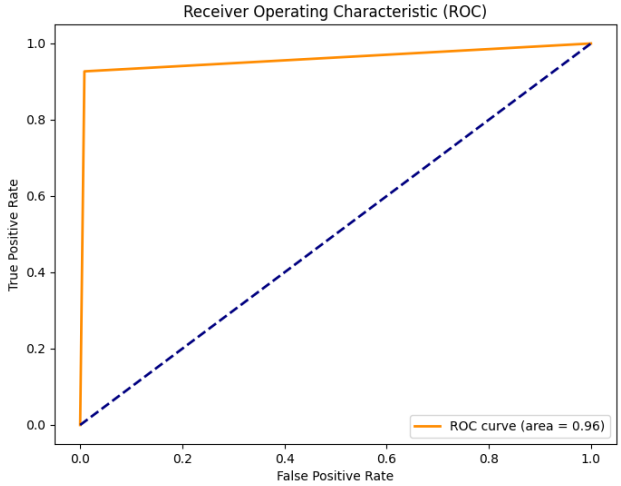


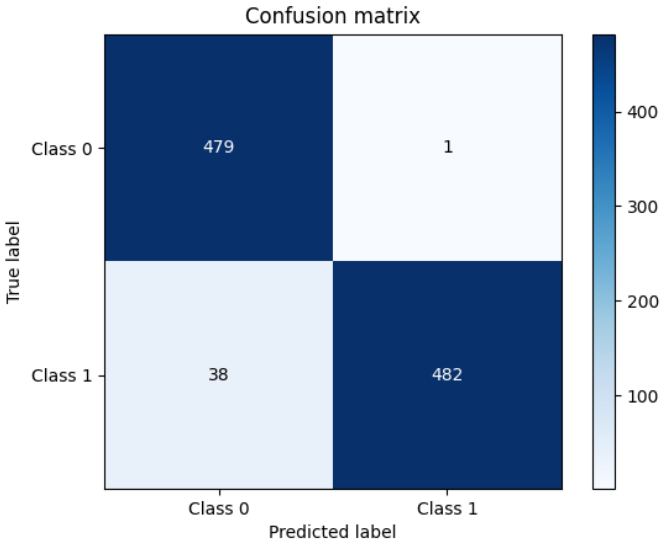
Below Table shows distribution of graphs and testing Accuracy for different size of testing dataset

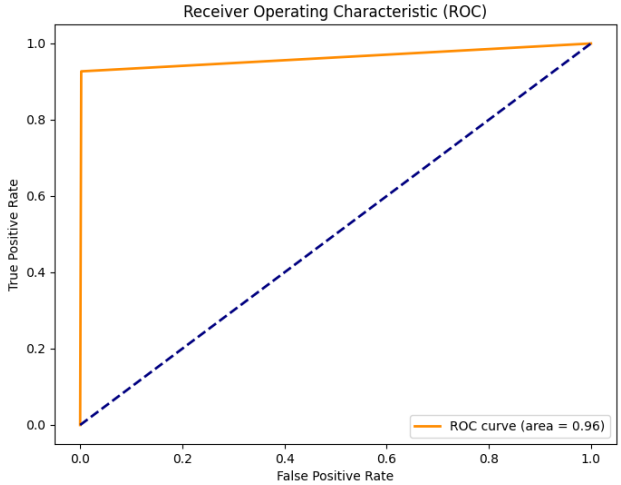
Total size of the Testing dataset	Barabasi Albert graphs	Erdos Renyi Graphs	Star graphs	wheel graph	cycle graph	Complete graph	Path graph	Test Accuracy
500	200	200	20	20	20	20	20	95.8%
1000	400	400	40	40	40	40	40	96.1%
1500	600	600	60	60	60	60	60	96.0%
2000	800	800	80	80	80	80	80	95.7%
2500	1000	1000	100	100	100	100	100	96.0%
3000	1200	1200	120	120	120	120	120	95.9%
3500	1400	1400	140	140	140	140	140	96.1%
4000	1600	1600	160	160	160	160	160	95.92%
4500	1800	1800	180	180	180	180	180	96.0%
5000	2000	2000	200	200	200	200	200	95.9%
5500	2200	2200	220	220	220	220	220	95.9%
6000	2400	2400	240	240	240	240	240	96.0%
6500	2600	2600	260	260	260	260	260	96.0%
7000	2800	2800	280	280	280	280	280	96.0%
7500	3000	3000	300	300	300	300	300	95.9%
8000	3200	3200	320	320	320	320	320	96.0%
8500	3400	3400	340	340	340	340	340	95.9%
9000	3600	3600	360	360	360	360	360	95.9%
9500	3800	3800	380	380	380	380	380	95.9%
10000	4000	4000	400	400	400	400	400	96.0%

- Below figures are the variations of 0 class labels and 1 class labels for different sizes of datasets.

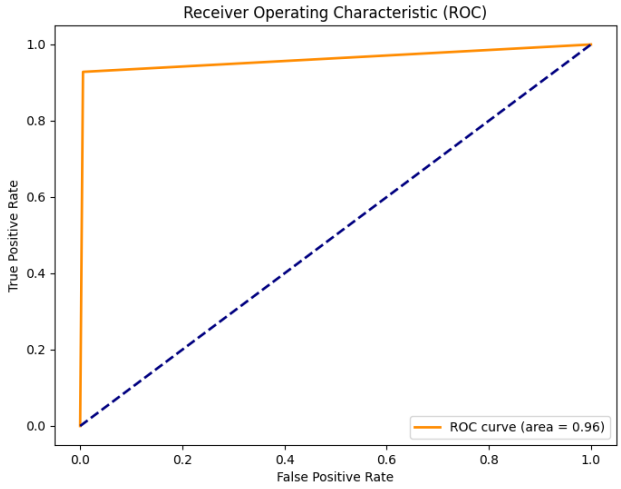


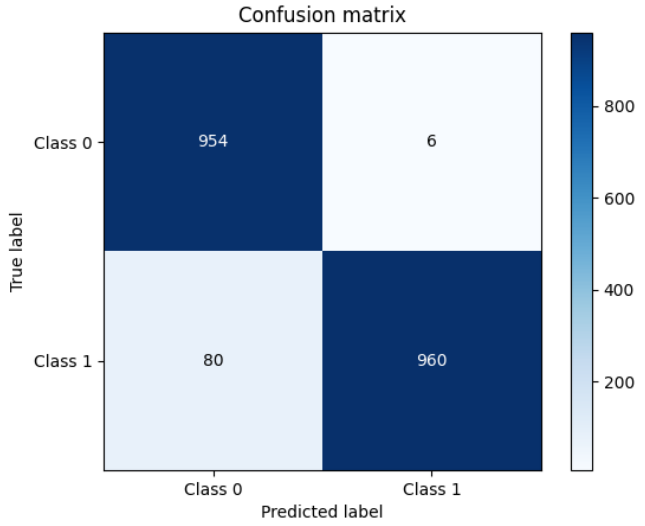


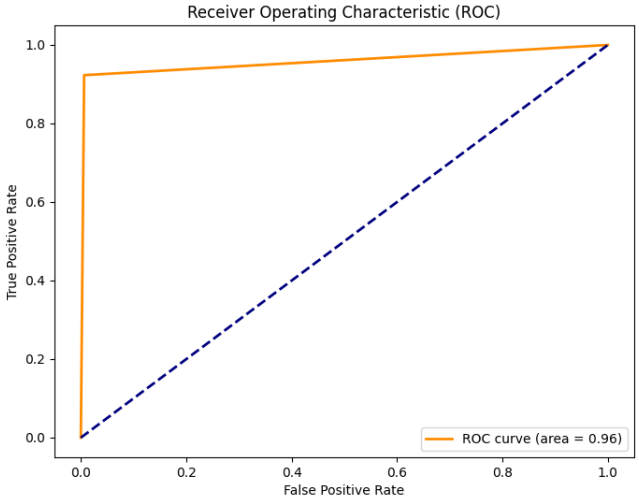


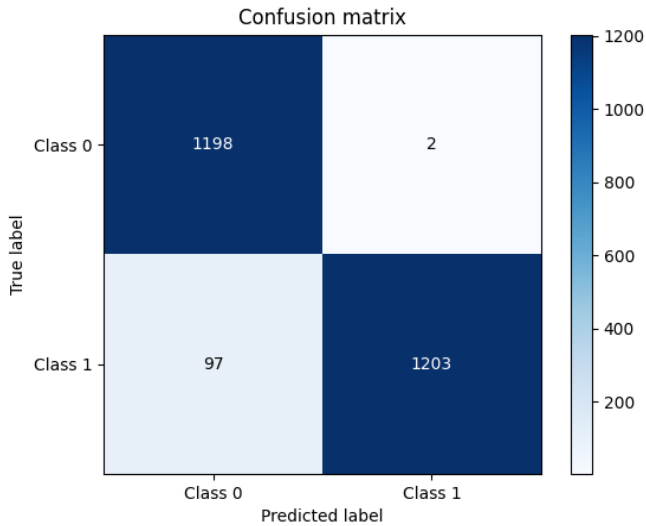


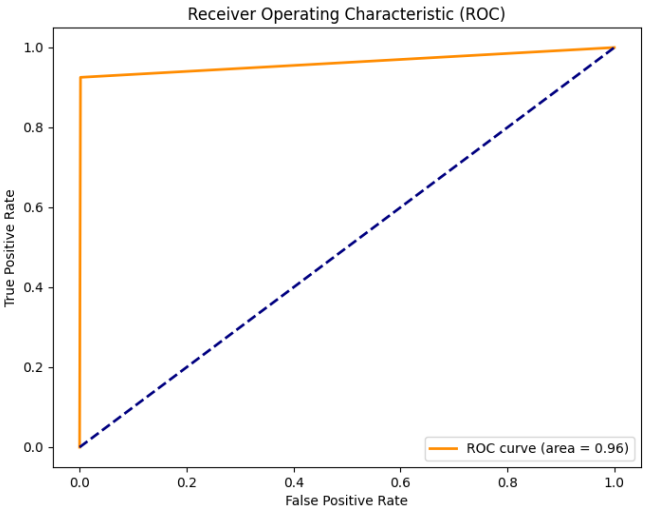

Confusion matrix
Class 0
Class 1
True label
716
4
56
724
Class 0
Class 1
Predicted label
700
600
500
400
300
200
100

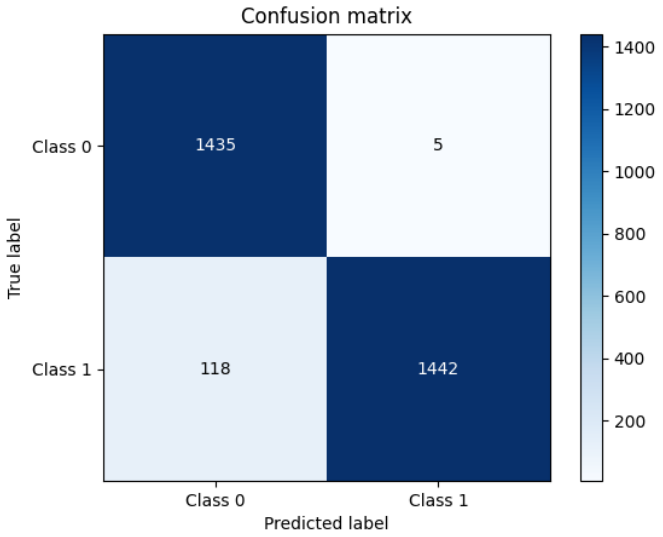

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

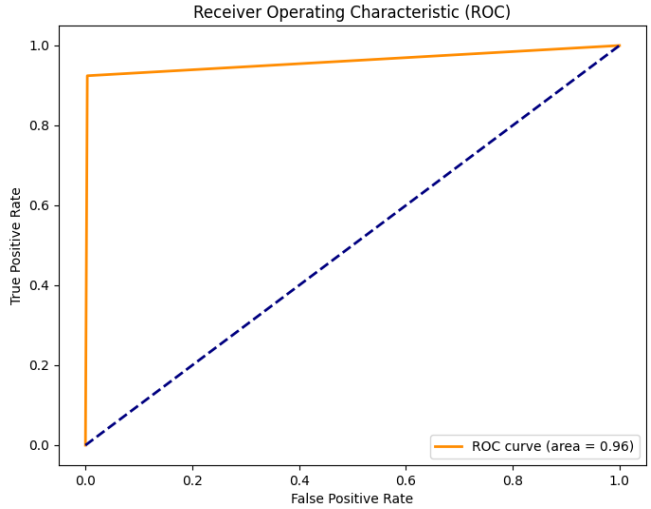

Confusion matrix
Class 0
Class 1
True label
954
6
80
960
Class 0
Class 1
Predicted label
800
600
400
200

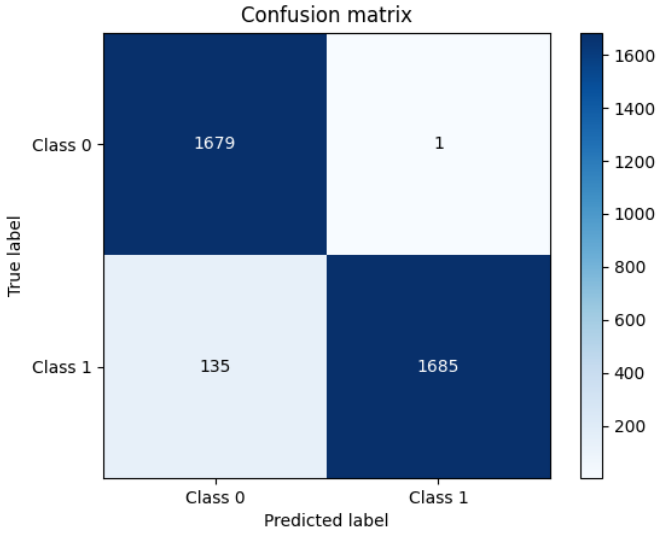

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

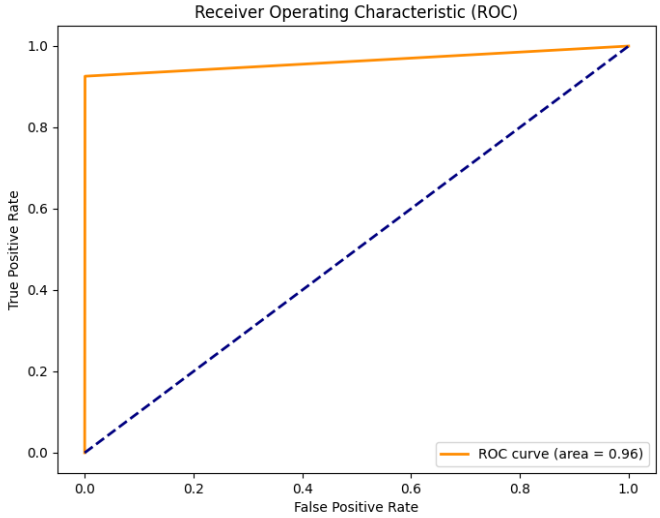

Confusion matrix
Class 0
Class 1
True label
1198
2
97
1203
Class 0
Class 1
Predicted label
1200
1000
800
600
400
200

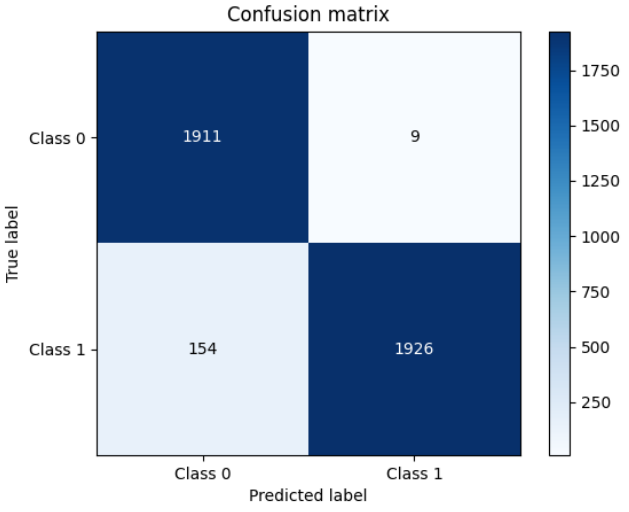

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

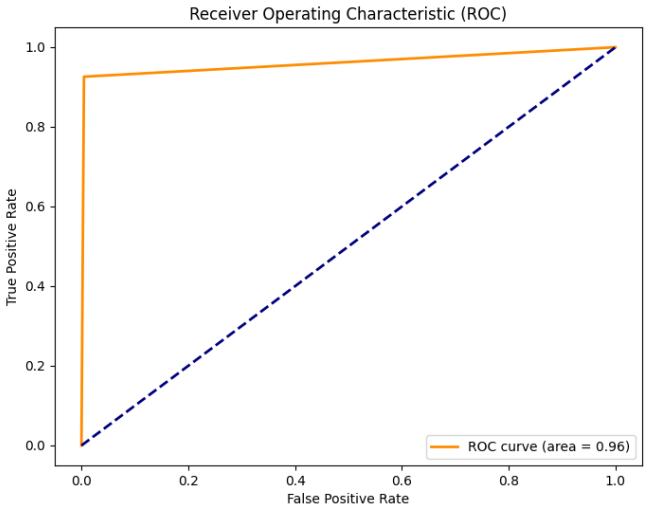

Confusion matrix
Class 0
Class 1
True label
1435
5
118
1442
Class 0
Class 1
Predicted label
1400
1200
1000
800
600
400
200

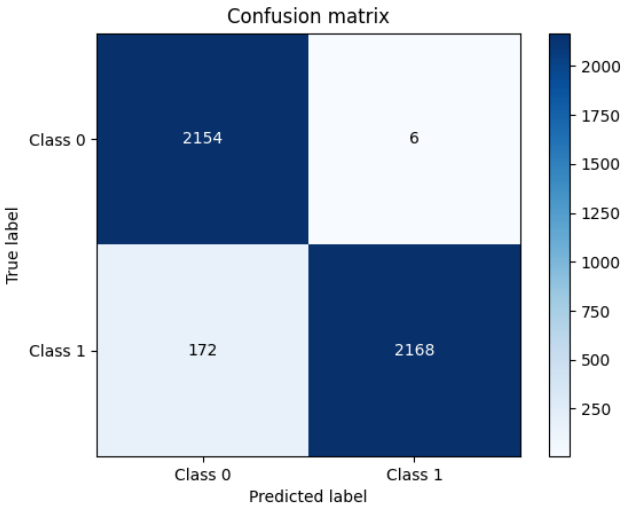

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

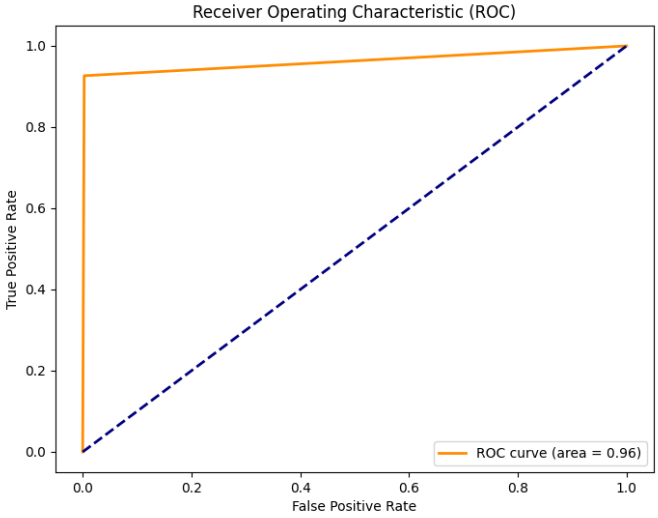

Confusion matrix
Class 0
Class 1
True label
1679
1
135
1685
Class 0
Class 1
Predicted label
1600
1400
1200
1000
800
600
400
200


Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

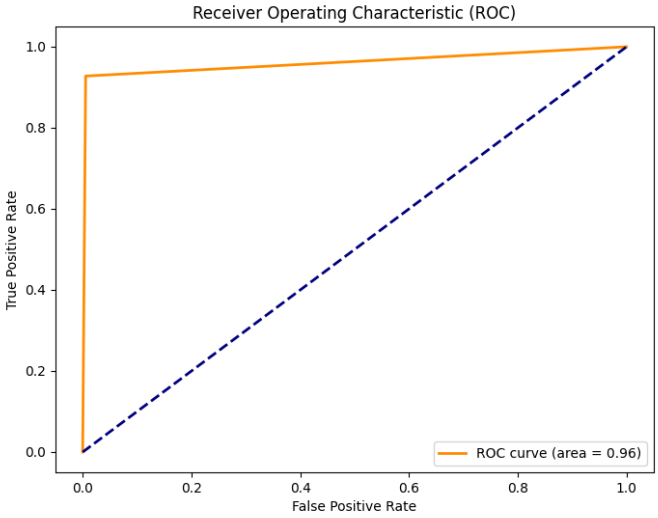

Confusion matrix
Class 0
Class 1
True label
1911
9
154
1926
Class 0
Class 1
Predicted label
1750
1500
1250
1000
750
500
250

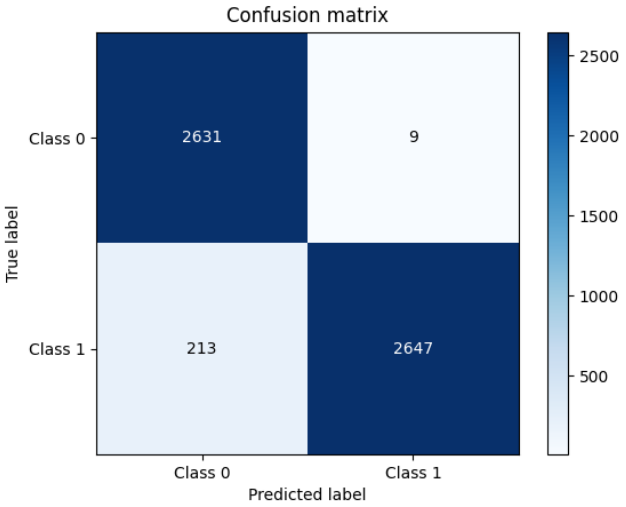

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

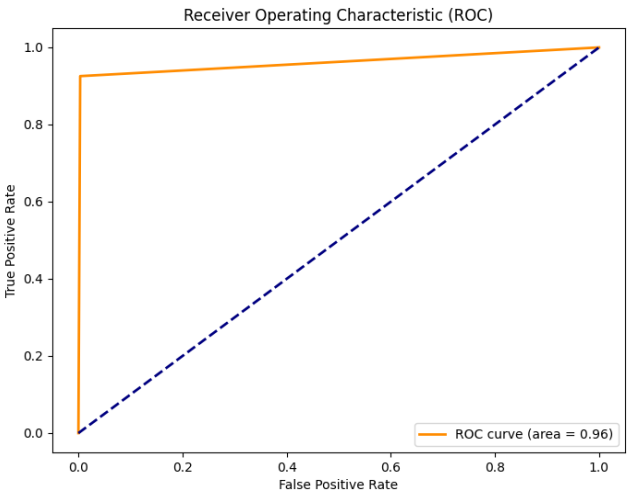

Confusion matrix
Class 0
Class 1
2154
6
172
2168
True label
Predicted label
2000
1750
1500
1250
1000
750
500
250


Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

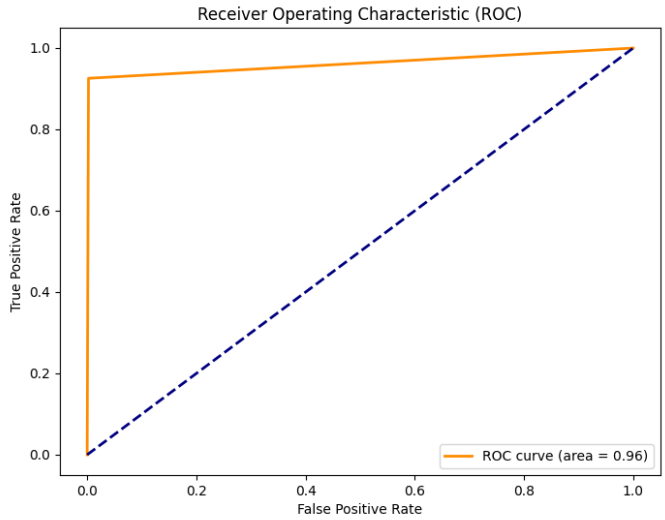

Confusion matrix
Class 0
Class 1
2387
13
188
2412
True label
Predicted label
2000
1500
1000
500

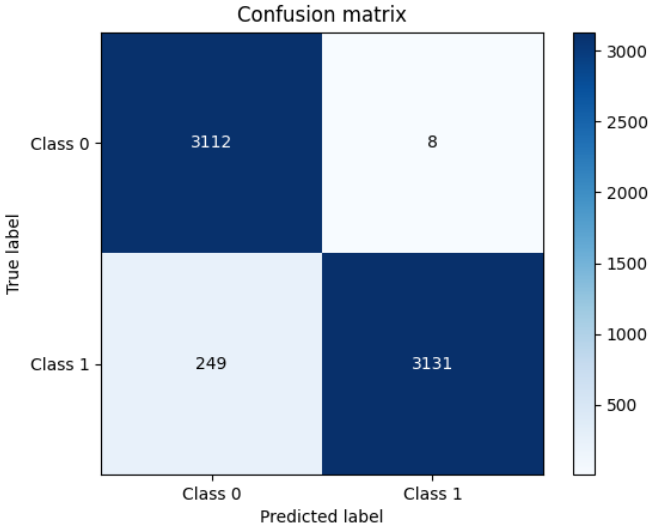

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

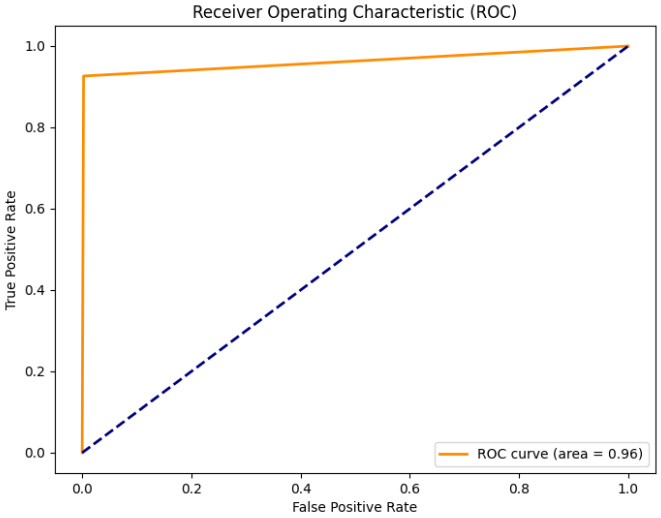

Confusion matrix
Class 0
Class 1
2631
9
213
2647
True label
Predicted label
2500
2000
1500
1000
500

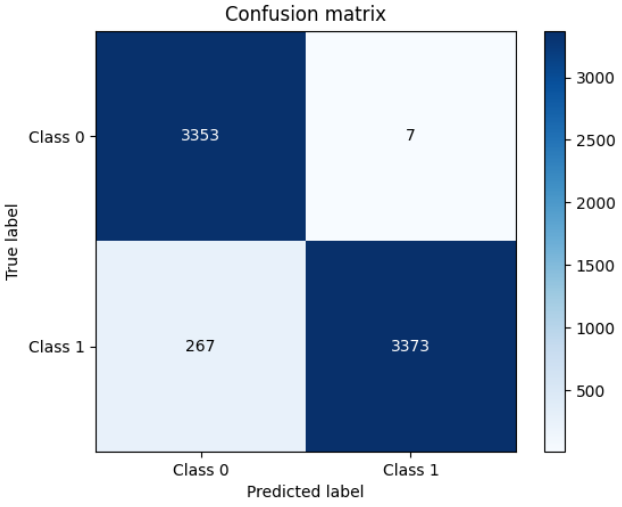

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)

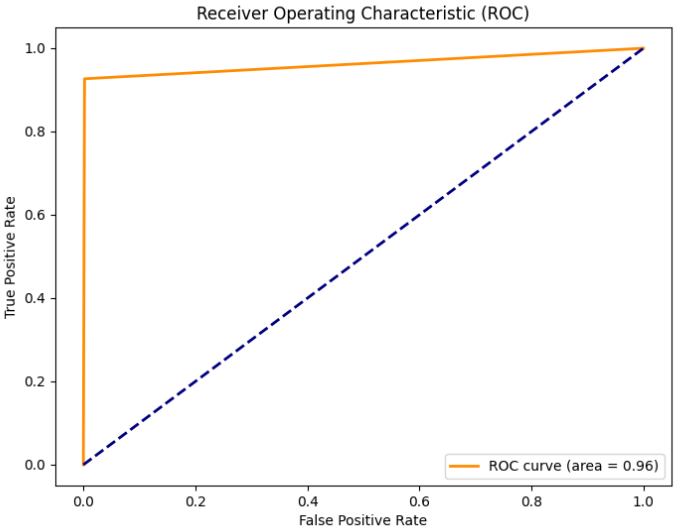

Confusion matrix
Class 0
Class 1
2873
7
232
2888
True label
Predicted label
Class 0
Class 1
2500
2000
1500
1000
500

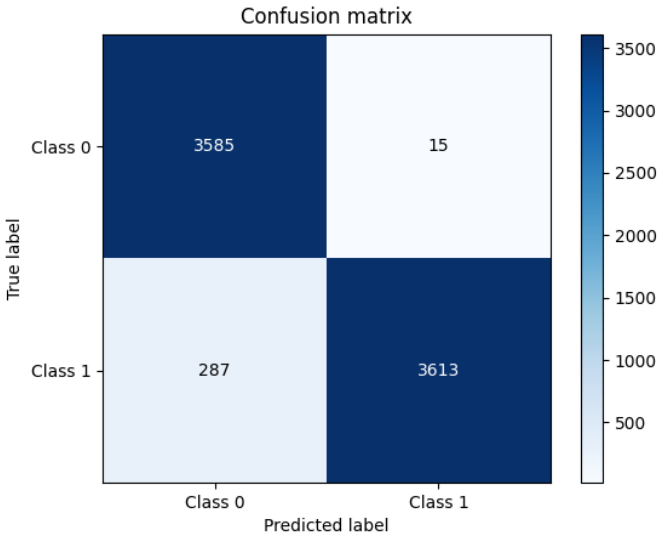

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

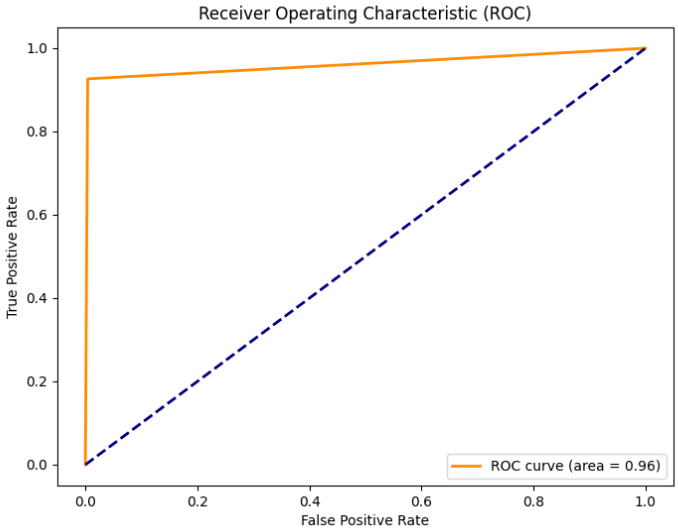

Confusion matrix
Class 0
Class 1
3112
8
249
3131
True label
Predicted label
Class 0
Class 1
3000
2500
2000
1500
1000
500

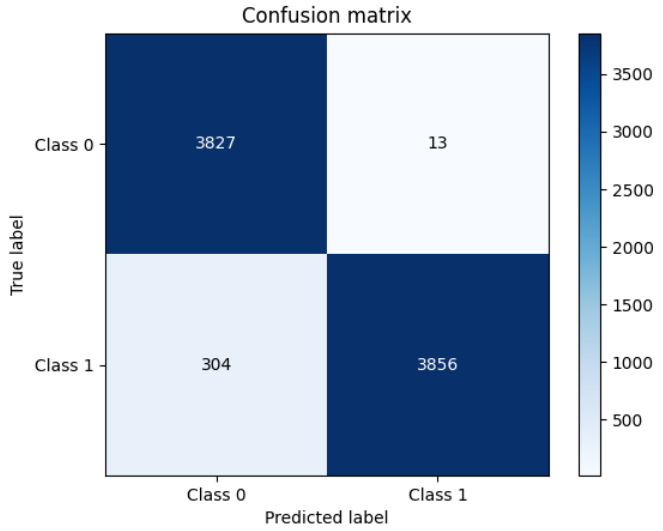

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

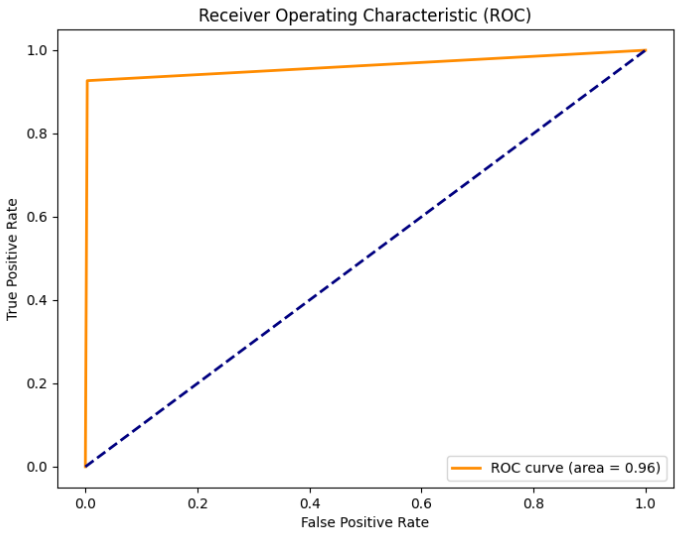

Confusion matrix
Class 0
Class 1
3353
7
267
3373
True label
Predicted label
Class 0
Class 1
3000
2500
2000
1500
1000
500

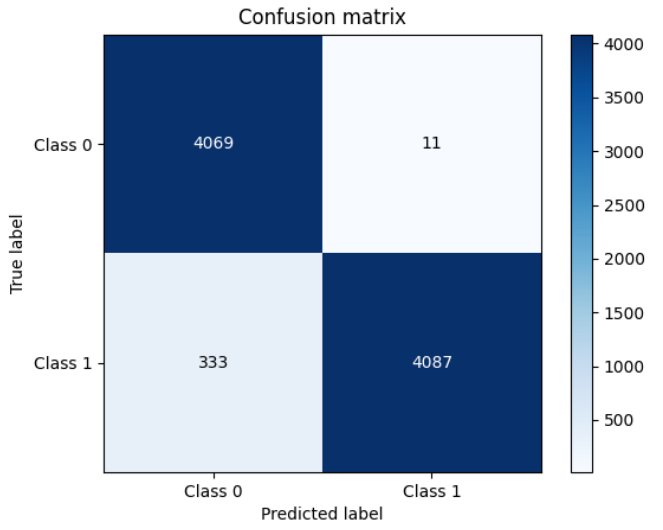

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

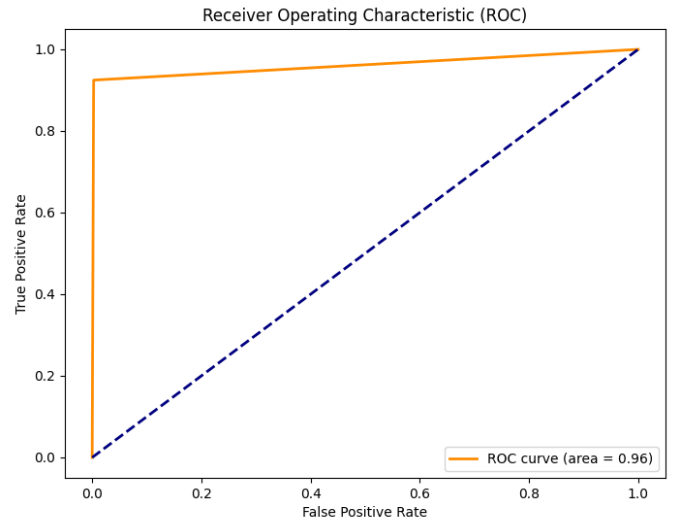

Confusion matrix
Class 0
Class 1
True label
3585
15
287
3613
Class 0
Class 1
Predicted label
3500
3000
2500
2000
1500
1000
500

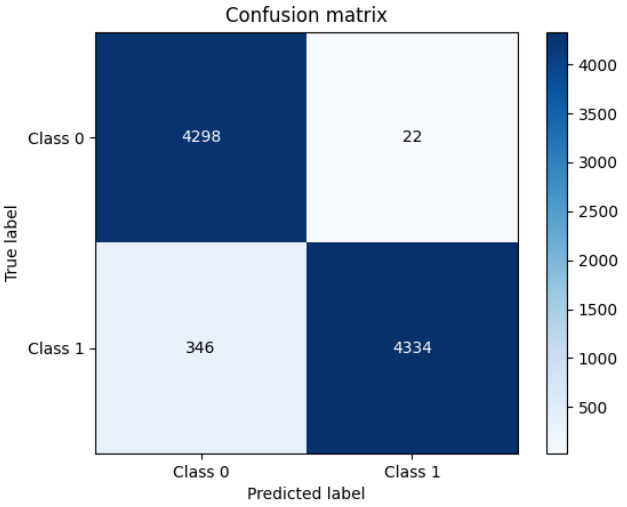

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

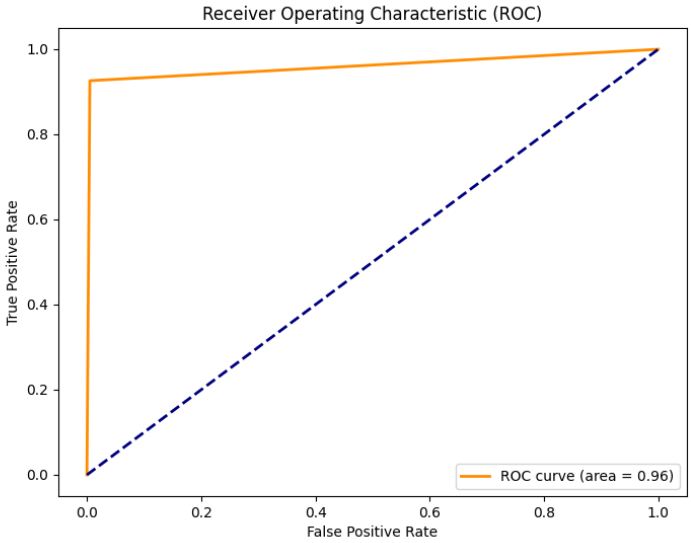

Confusion matrix
Class 0
Class 1
True label
3827
13
304
3856
Class 0
Class 1
Predicted label
3500
3000
2500
2000
1500
1000
500

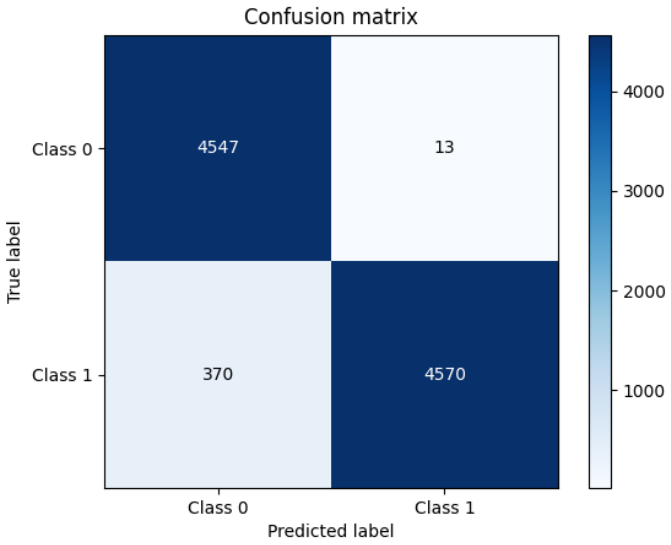

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

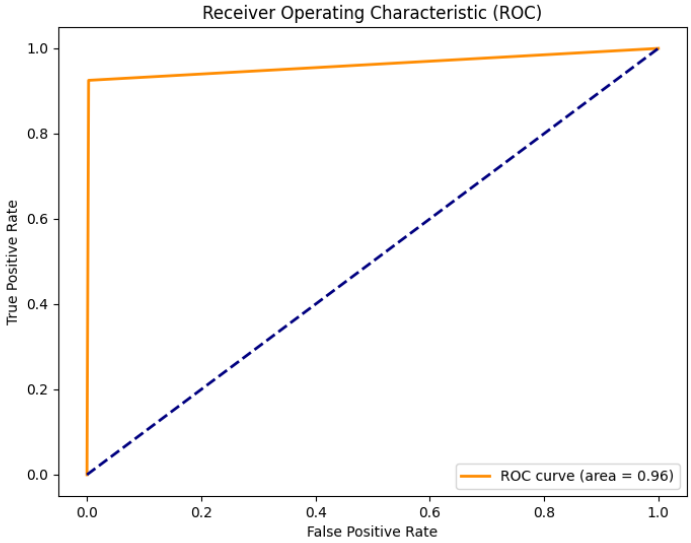

Confusion matrix
Class 0
Class 1
True label
4069
11
333
4087
Class 0
Class 1
Predicted label
4000
3500
3000
2500
2000
1500
1000
500

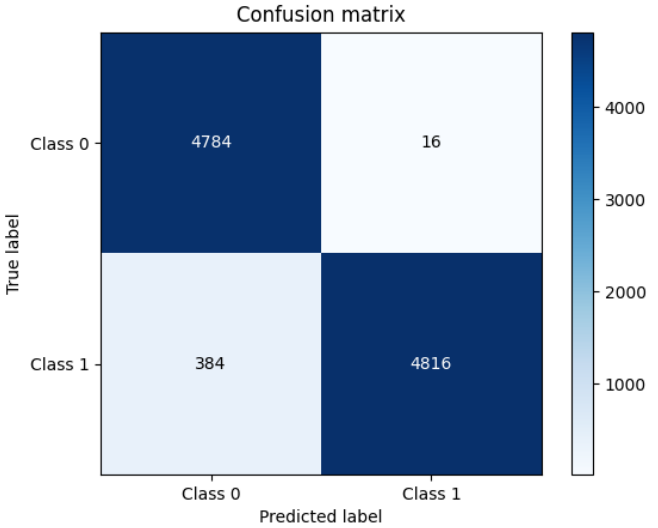

Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

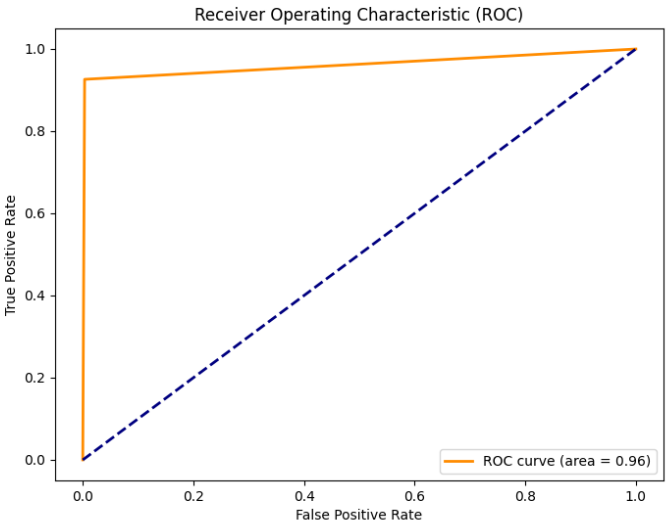

Confusion matrix
Class 0
Class 1
4298
22
346
4334
True label
Predicted label
4000
3500
3000
2500
2000
1500
1000
500


Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

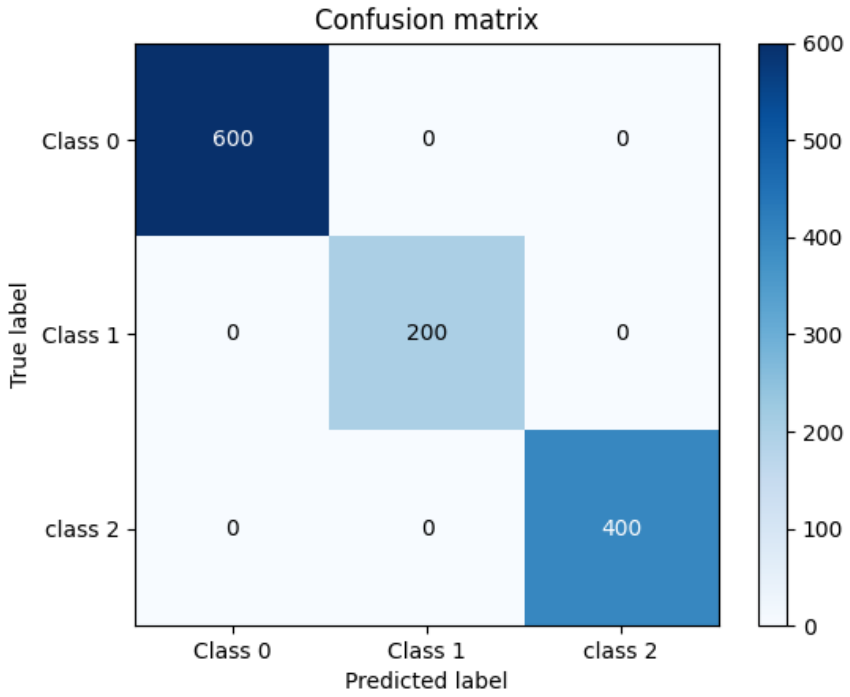

Confusion matrix
Class 0
Class 1
4547
13
370
4570
True label
Predicted label
4000
3000
2000
1000


Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0


Confusion matrix
Class 0
Class 1
4784
16
384
4816
True label
Predicted label
4000
3000
2000
1000


Receiver Operating Characteristic (ROC)
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

- We used Multiclass labels and tested using this dataset and results are accurate and all labels are Hitting.



6.Results

6.1 Datasets

TUDataset is a collection of benchmark datasets for graph machine learning and graph neural network (GNN) learning with classifications.It offers a standardized method to assess how well GNN models perform on a variety of graph categorization tasks.It is frequently used as a benchmark in the research community to assess how well GNNs perform on various tasks using graphs.

This dataset, which covers a variety of areas including social networks, bioinformatics, and chemistry, offers a broad collection of datasets with various graph shapes, sizes, and features.These datasets allow us to compare the outcomes of various GNN models, assess how well they function, and create new graph-based machine learning techniques.

- MUTAG Dataset
- NCI1 Dataset
- PROTEINS Dataset
- ENZYMES Dataset
- MCF-7 Dataset
- YEAST Dataset
- PCT_FM Dataset
- NCI109 Dataset

The TUDataset library offers a Python API that makes it simple to access and load datasets, as well as methods for processing and analyzing graph data. It also assists in dividing the graph data into training and testing sets, making it easier to use.

- from torch_geometric.datasets import TUDataset
- from torch.nn import Linear
- from torch_geometric.nn import GCNConv, global_mean_pool

- from torch_geometric.data import Data, Dataset
- from torch_geometric.loader import DataLoader

6.2 Dataset statistics :

DATASET	Number of Graphs	Number of Features	Training Set	Testing Set	Training Accuracy	Testing Accuracy
Mutag	188	7	150	38	88.0%	73.6%
NCI1	4,110	37	3,300	810	68.5%	67.2%
PROTEINS	1,113	3	888	225	71.9%	73.3%
ENZYMES	600	3	480	120	37.0%	33.3%
MCF-7	27,770	46	22,000	5,770	91.8%	91.4%
YEAST	79,601	74	64,000	15,601	88.1%	88.3%
PTC_FM	349	18	280	69	72.1%	65.2%
NCI109	4,127	38	3,300	827	67.4%	62.9%
MY_DATASET	5000	10	4000	1000	96.0%	93.0%

6.3 Mutag dataset :

The MUTAG dataset is a collection of 188 chemical compounds that are divided into two classes based on their mutagenic effect on a bacterium.

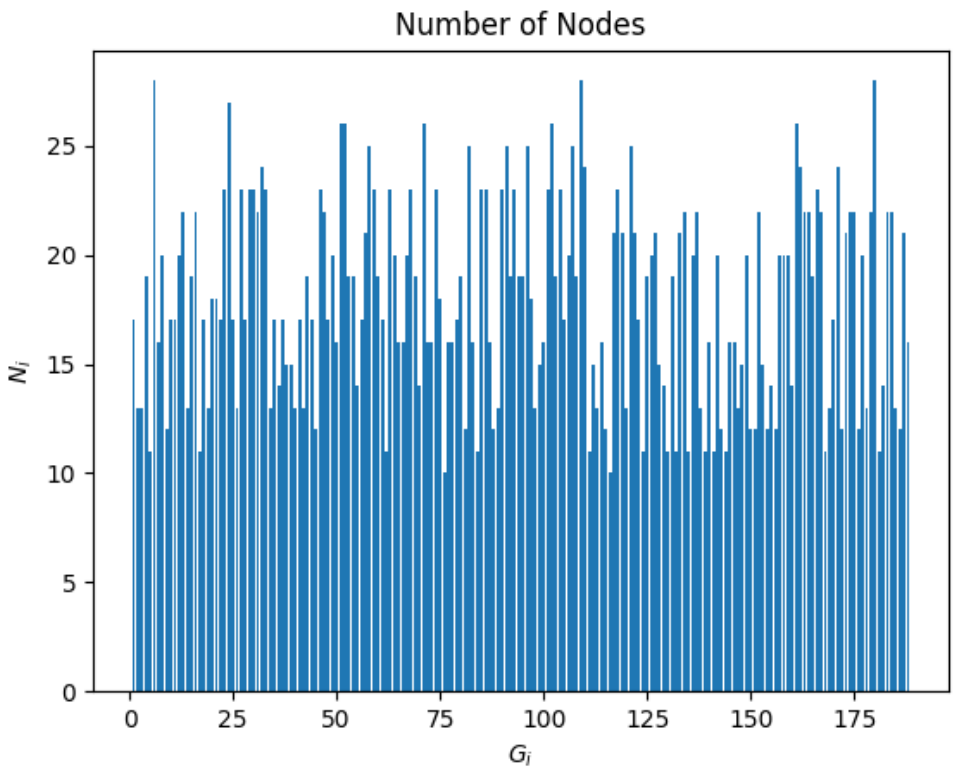
- Chemical data is represented as graphs
- Nodes represent atoms
- Edges represent chemical bonds

The goal of the GNN research on this dataset is typically to train a model to predict the mutagenic class of chemical compounds based on their graph representation.

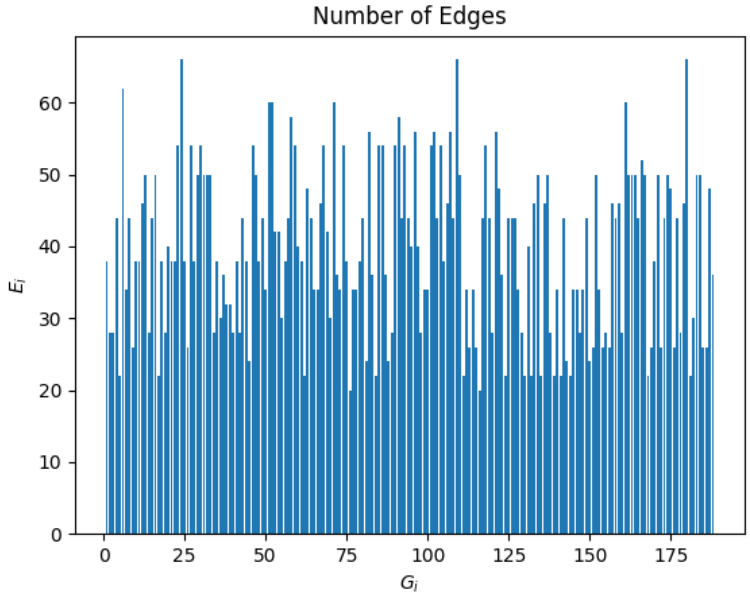
MUTAG dataset is loaded into a GNN model, the model is trained using Graph Convolutional Network, Graph Neural Network and so on. The goal is to learn the patterns or representations from the graph data that can effectively predict the mutagenic class of the chemical compounds such as mutagenic (1) or non-mutagenic (0), based on the graph structure and associated node/edge features.

6.3.1 Analysis:

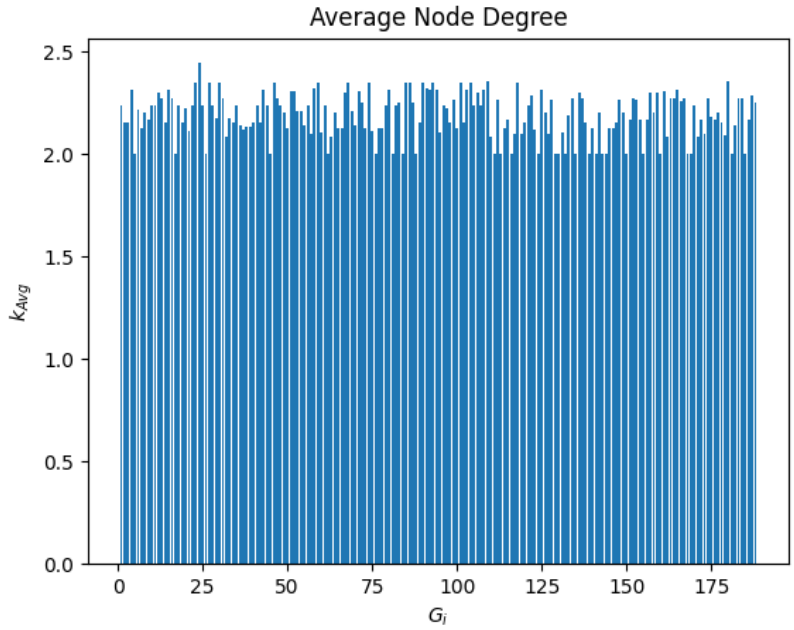
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



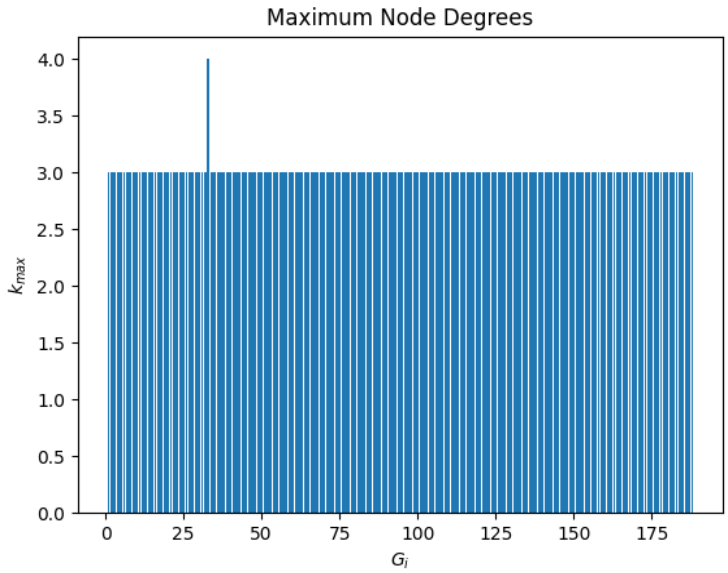
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



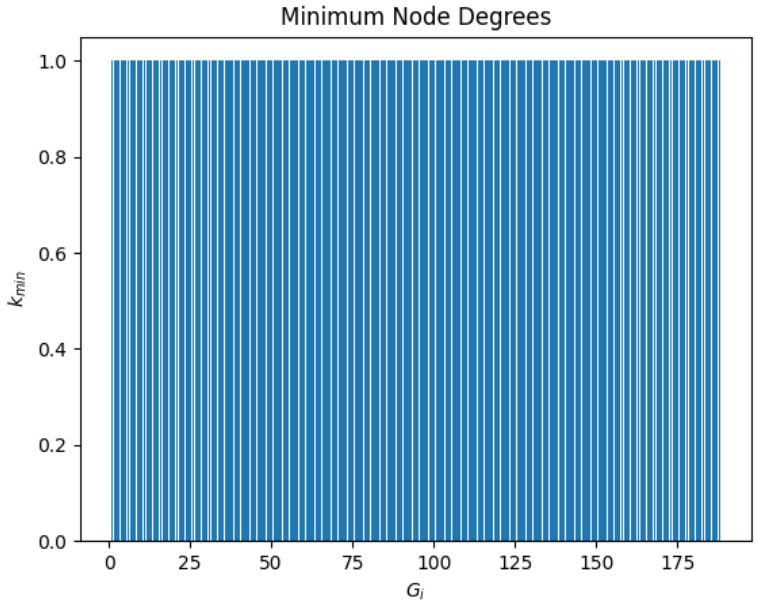
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



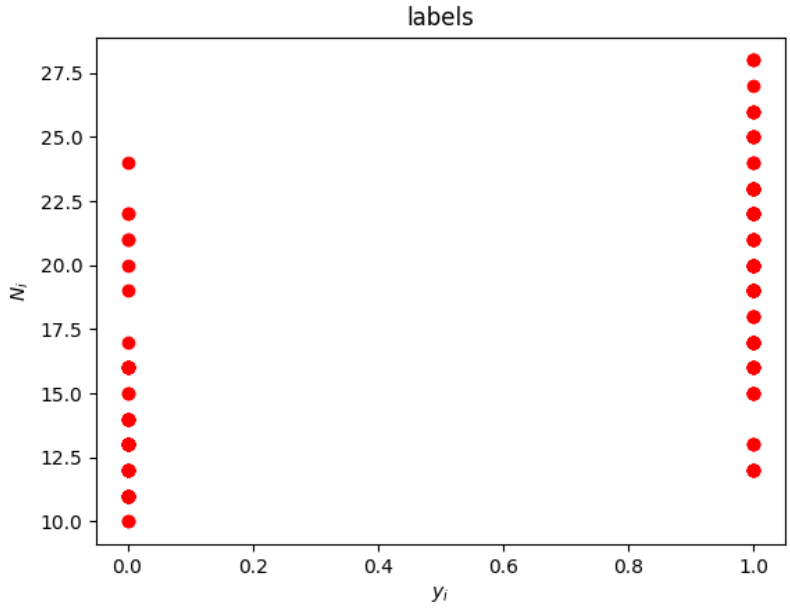
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



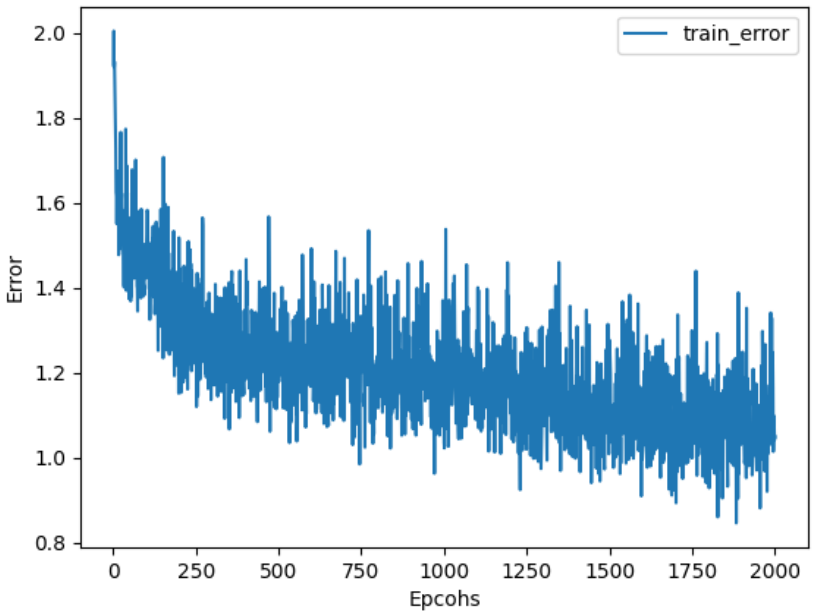
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



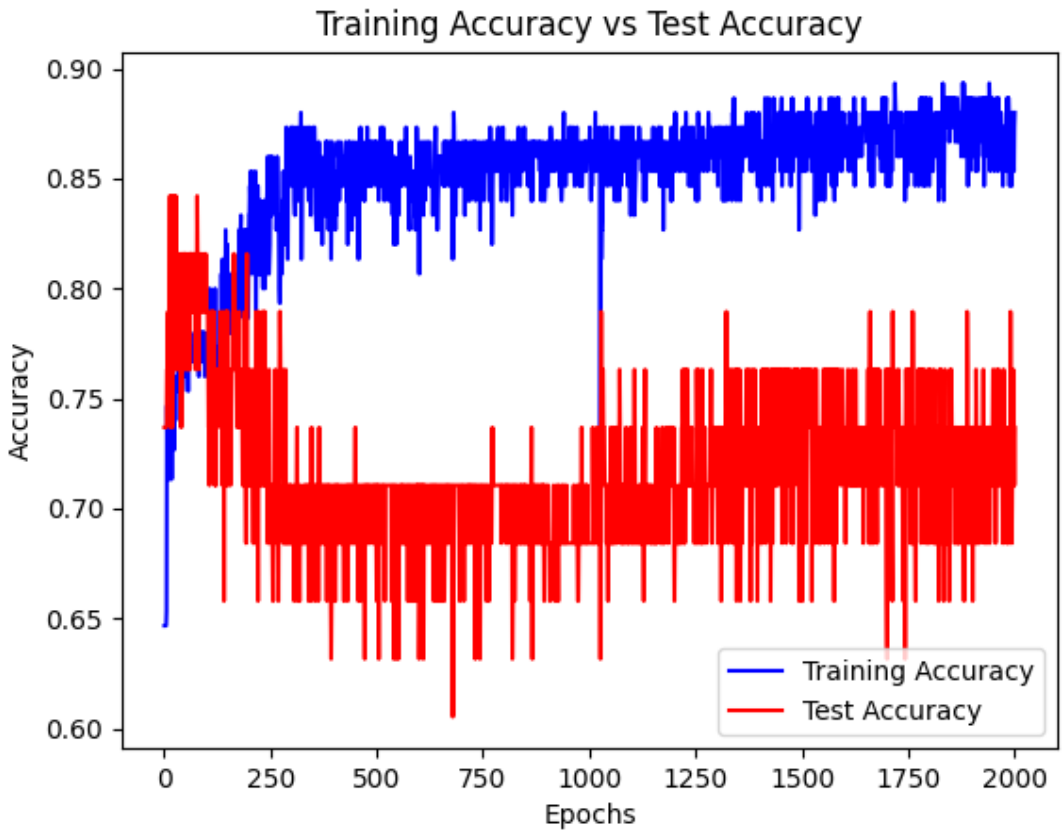
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



6.4 NCI1 Dataset :

The NCI1 dataset is a collection of 4110 chemical compounds that come under the cheminformatics domain that are divided into two classes based on the anti-cancer screens where the chemicals are assessed as positive or negative to cell lung cancer.

- Chemical data is represented as graphs
- Nodes represent atoms
- Edges represent chemical bonds

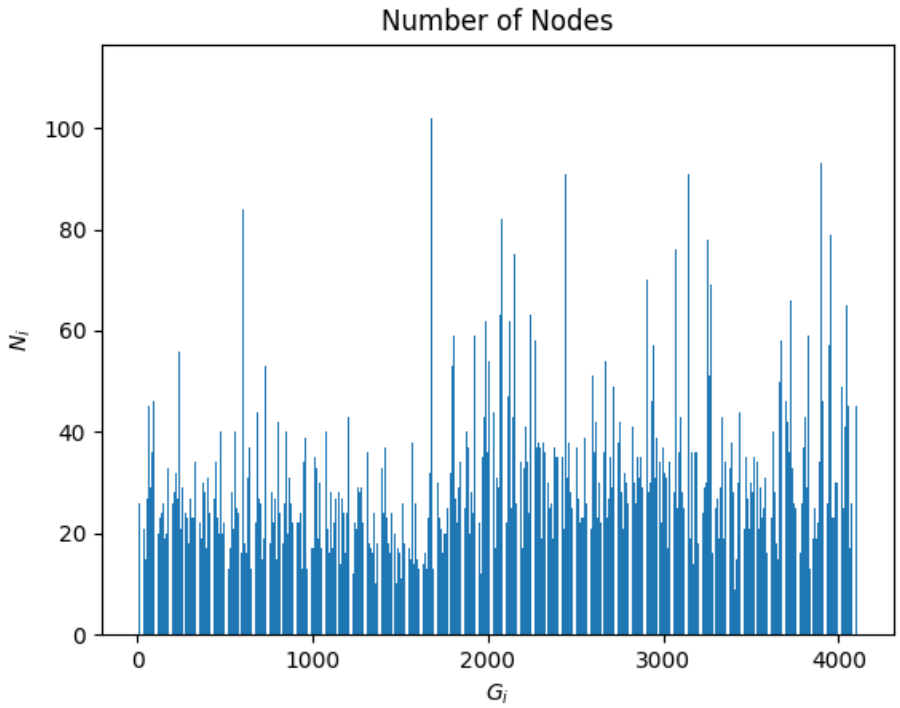
The NCI1 dataset is a binary classification dataset, where compounds are labeled as either 0 or 1. The label "1" indicates that the compound has anti-cancer activity, while the label "0" indicates that the compound does not have anti-cancer activity.

The NCI1 dataset consists of a total of 4,110 compounds, which are divided into 3,500 compounds for training and 610 compounds for testing and it is mainly used for graph classification, chemical property prediction.

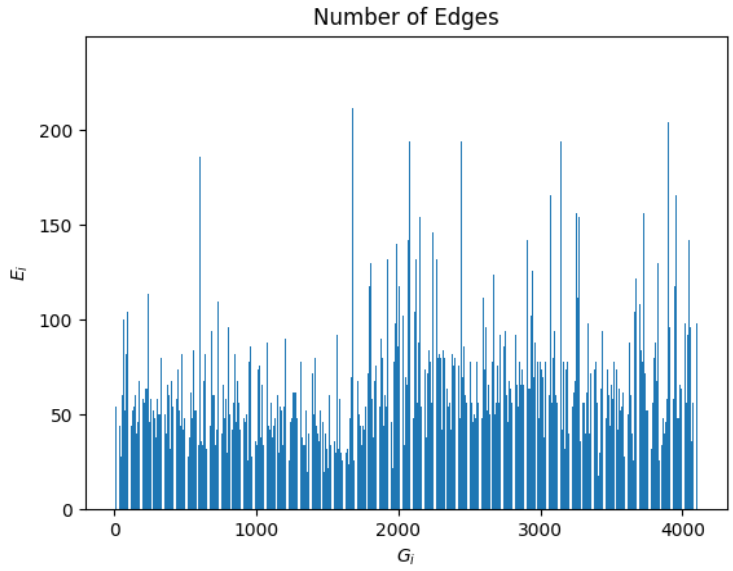
The NCI1 dataset is the complex and variable nature of chemical compounds, which makes it difficult to accurately predict their anti-cancer activity. Compounds in the dataset can have different sizes, atom types, and bond types so that model has to learn more inorder to produce good results.

6.4.1 Analysis:

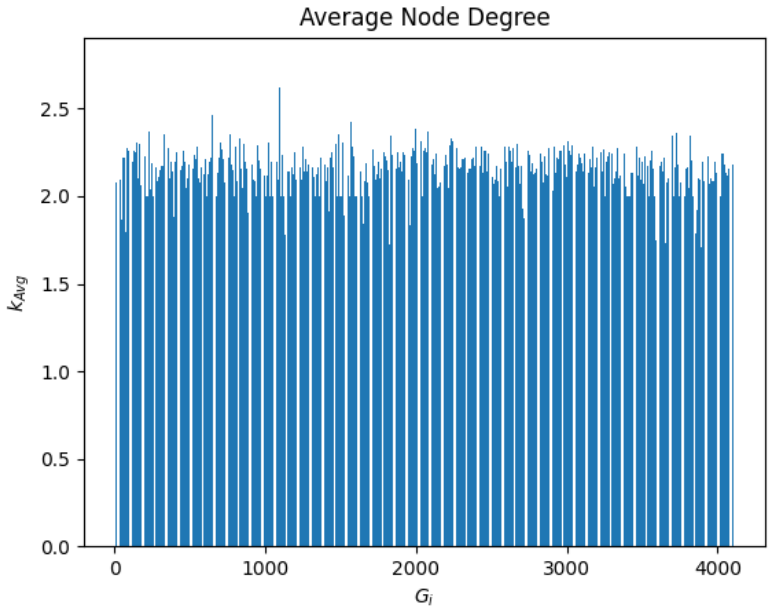
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



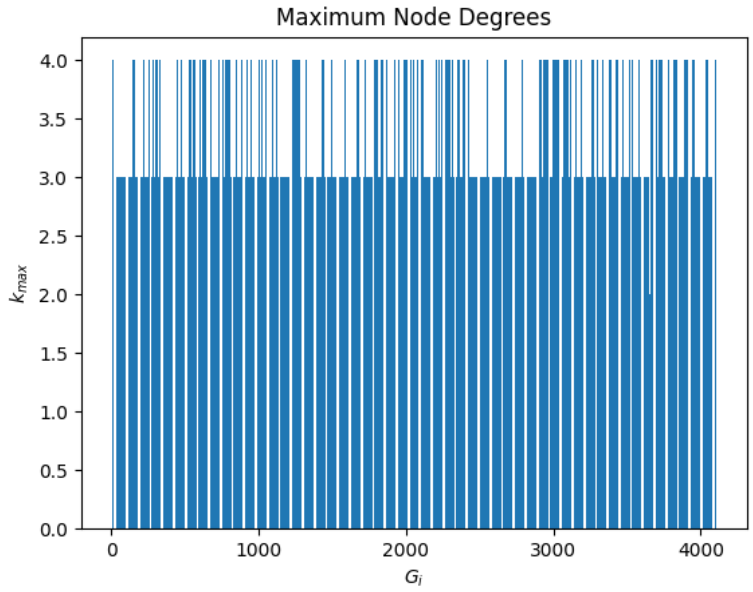
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



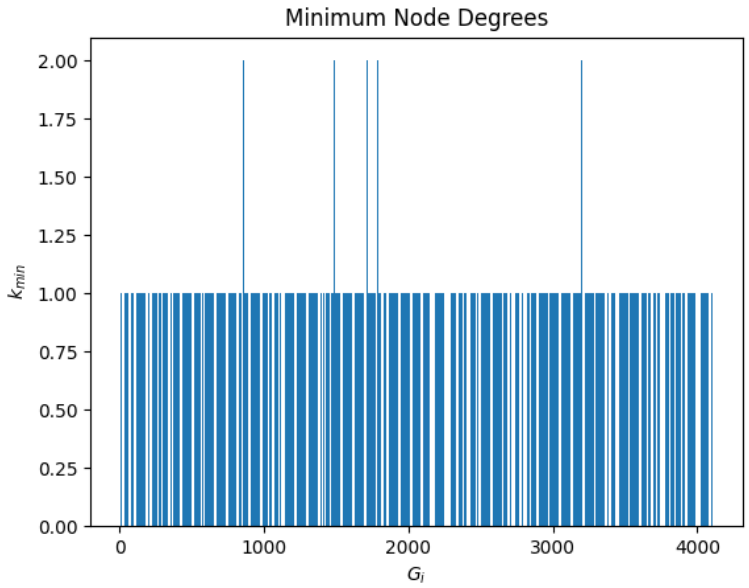
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



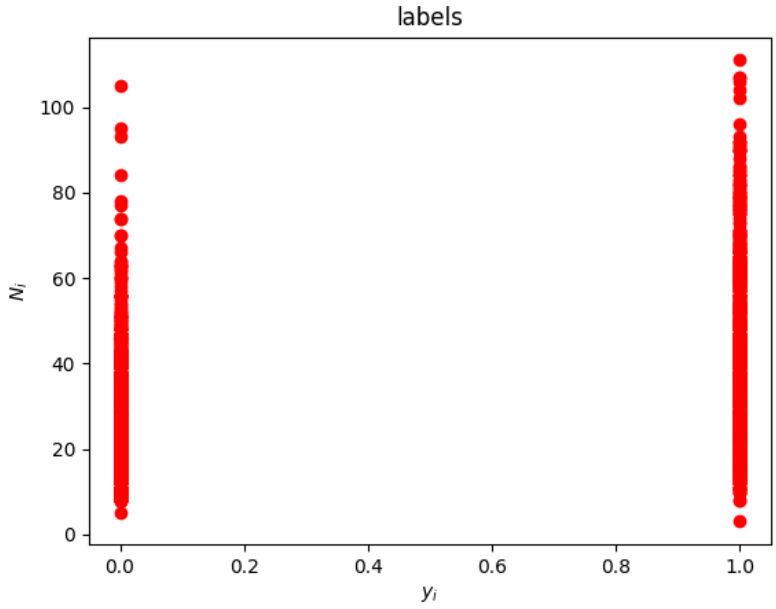
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



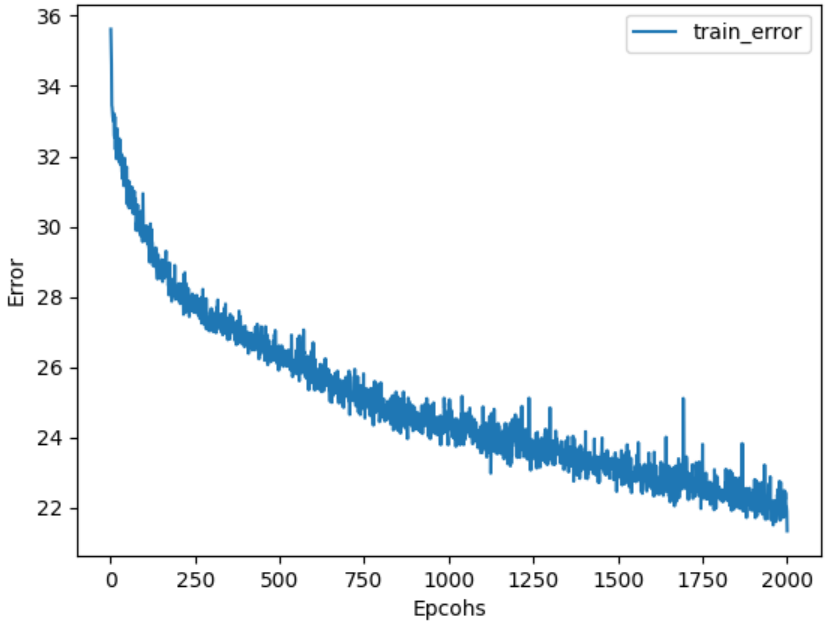
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



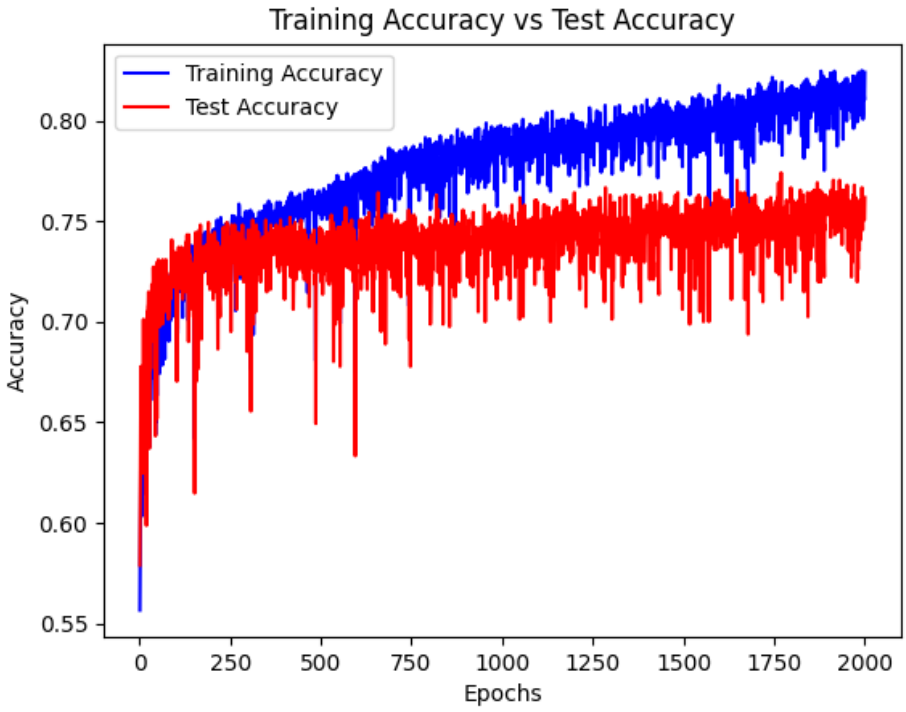
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



6.5 PROTEINS Dataset :

Proteins is a dataset of proteins that are classified as enzymes or non-enzyme that are divided into two classes based on Angstroms.These datasets are widely used in bioinformatics and computational biology to study protein structure, function, interactions, and evolution.

- Chemical data is represented as graphs(1113)
- Nodes represent amino acids
- Edges represent bonds
- Node features: Atom-level features such as element type

The Proteins dataset is a binary classification dataset,where the label indicates whether the protein is an enzyme or not.

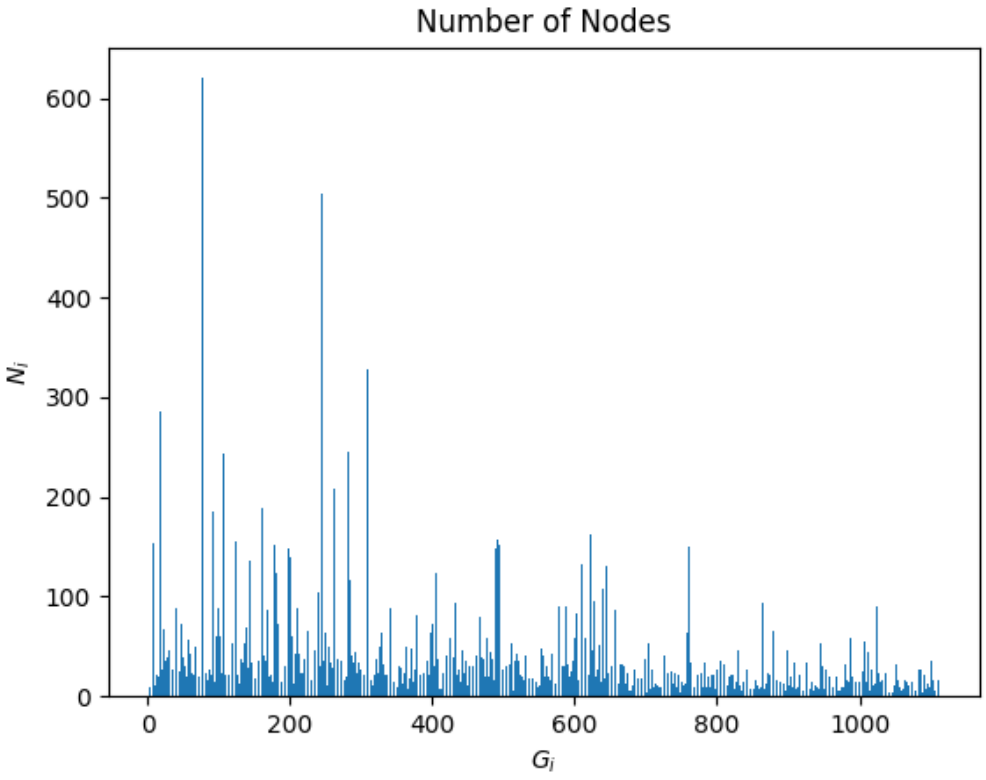
Enzyme : Graphs with label ‘1’ represent proteins that are enzymes and these proteins do possess enzymatic activity.

Non-Enzyme : Graphs with label ‘0’ represent proteins that are not enzymes and these proteins do not possess enzymatic activity and are typically involved in other cellular functions or structural roles within cells.

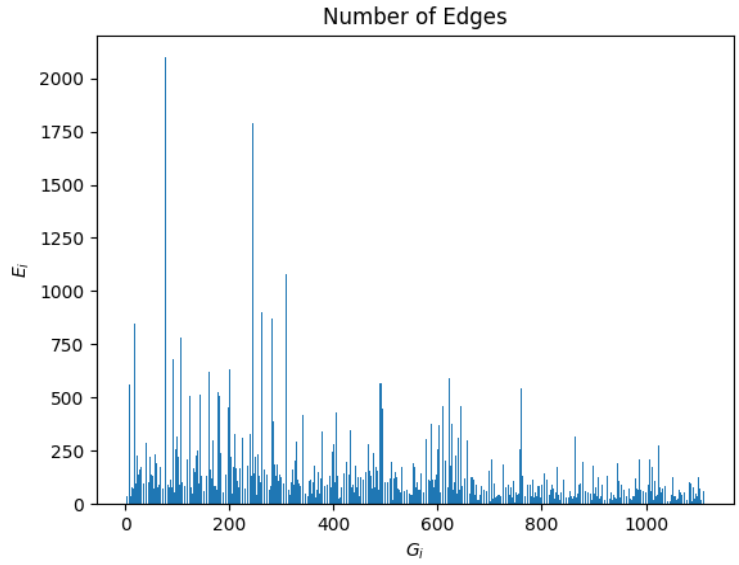
Proteins dataset is used for tasks such as protein classification, enzyme prediction, and other protein-related analysis tasks using graph-based machine learning models, such as graph neural networks (GNNs).

6.5.1 Analysis:

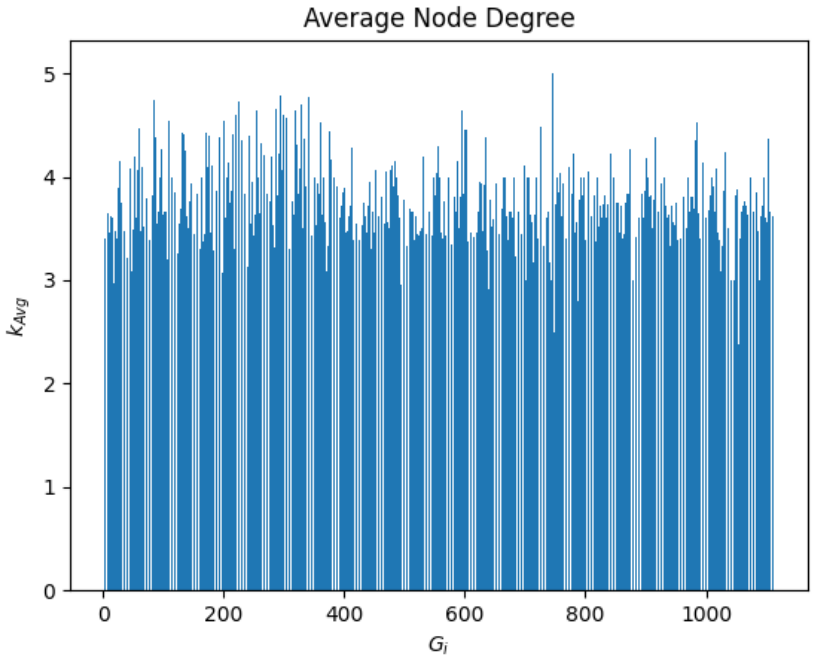
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



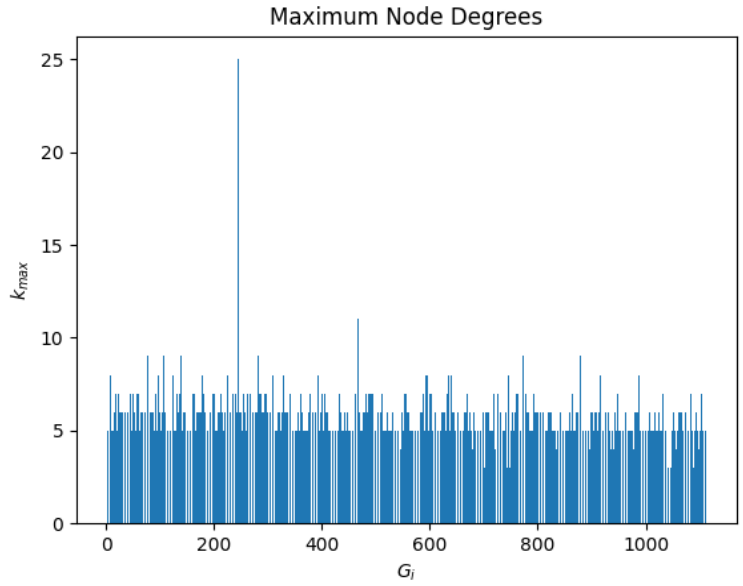
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



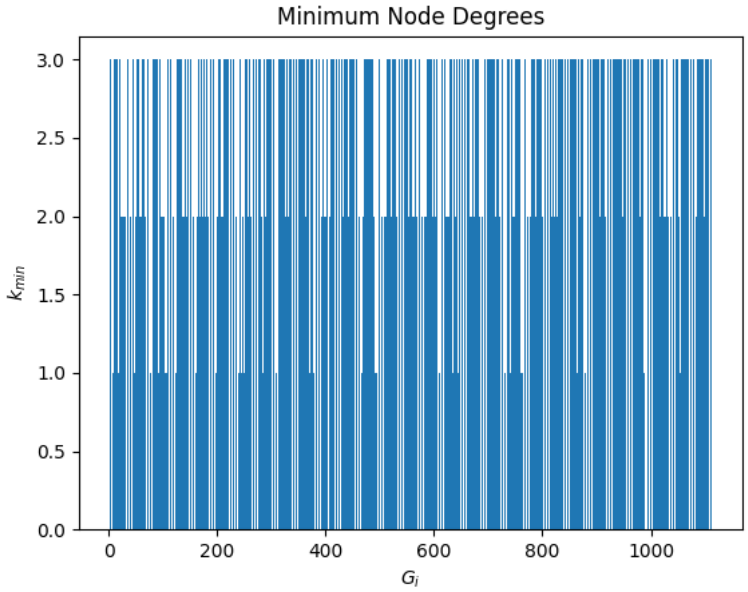
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



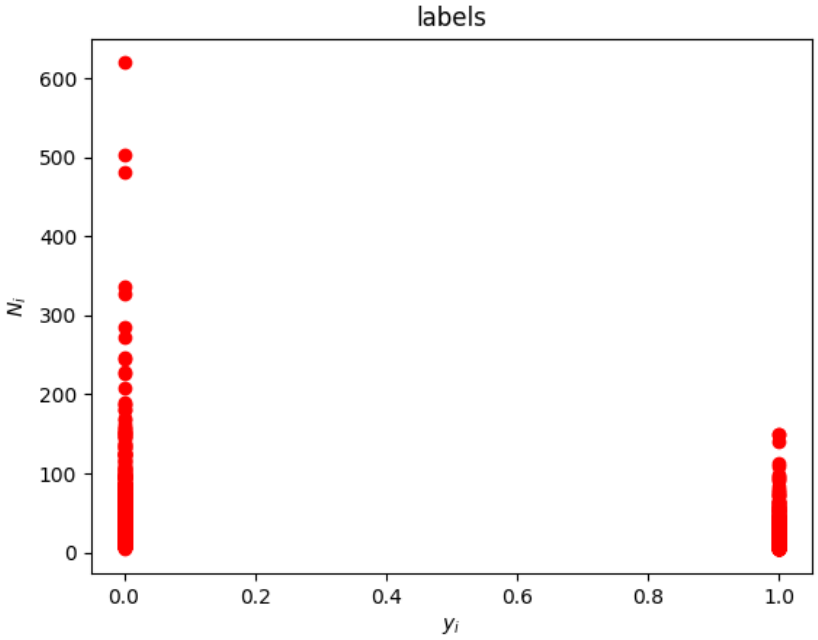
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



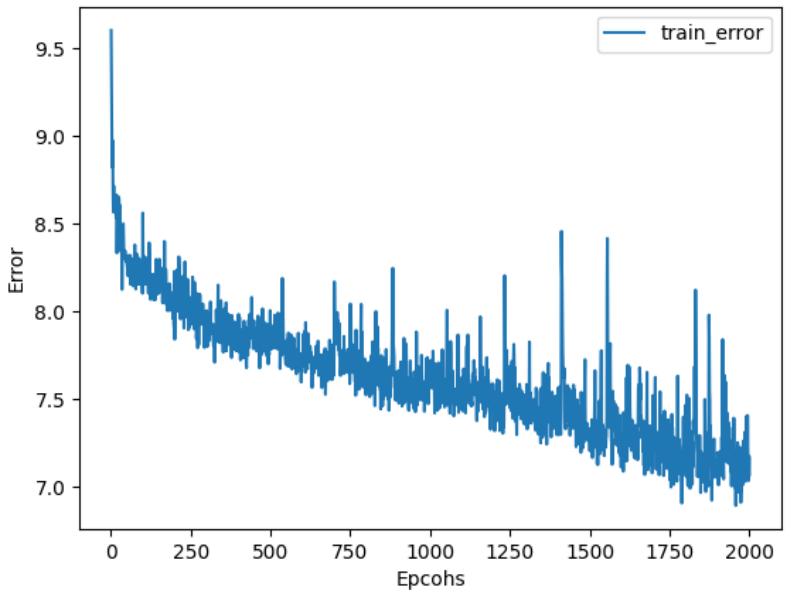
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



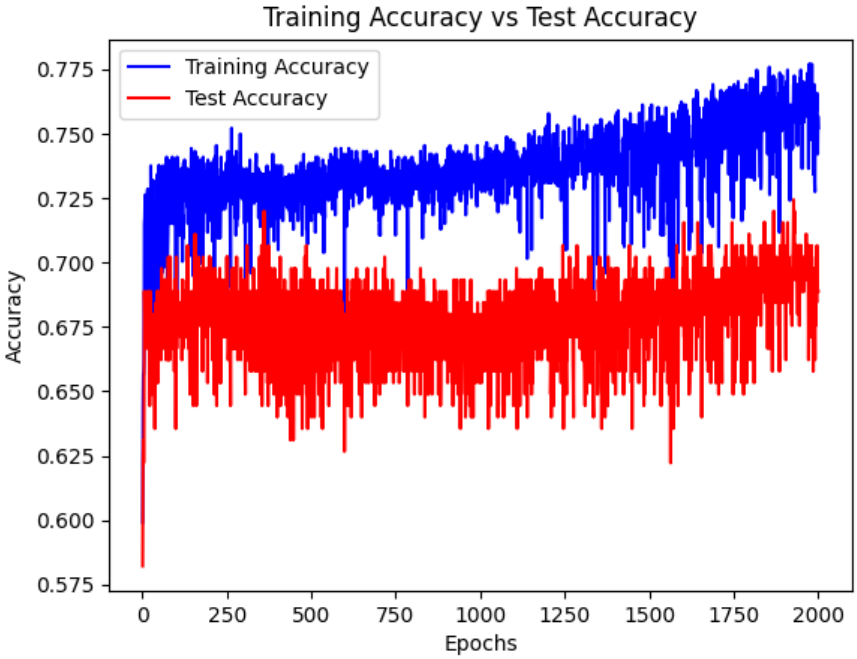
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



6.6 ENZYMES Dataset :

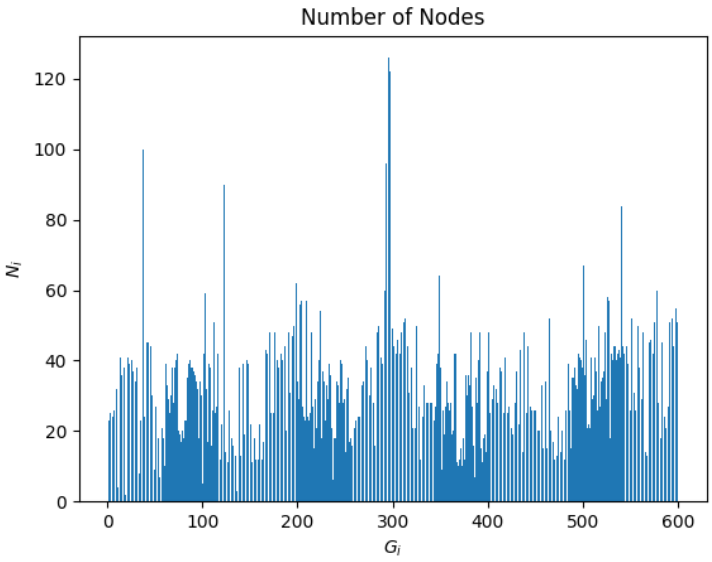
The Enzymes dataset is a collection of '600' enzyme graphs, where each graph represents the structural information of an enzyme molecule. Enzymes are biological molecules that function as catalysts in living organisms.

- Graphs represent molecular structure of enzyme
- Nodes represent atom
- Edges represent chemical bond
- Node features: Atom-level features such as element type

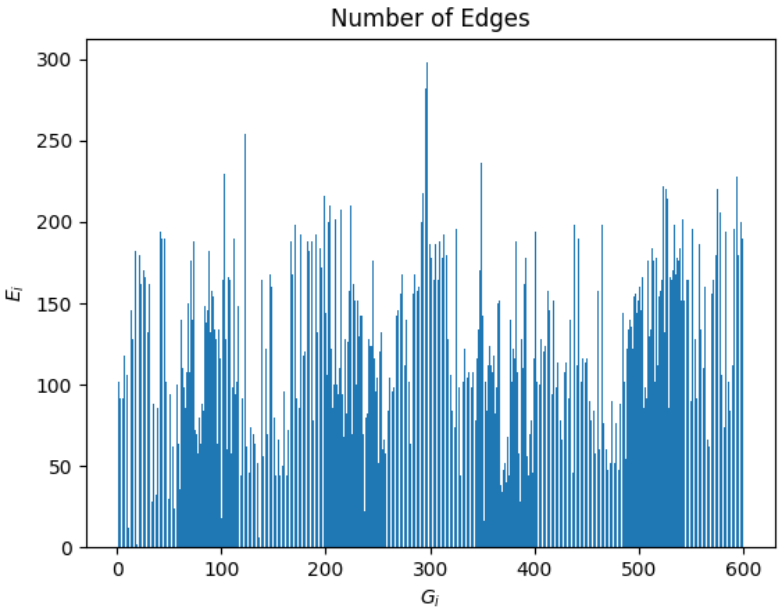
The Enzymes dataset is used for classification tasks where the goal is to predict the functional class or family of enzymes based on their molecular structure.Dataset typically includes labels for each graph.These labels are used as the target labels(6) for the classification task, and the goal is to train a deep learning model that can accurately predict the functional class or family of enzymes based on their molecular structure.

6.6.1 Analysis:

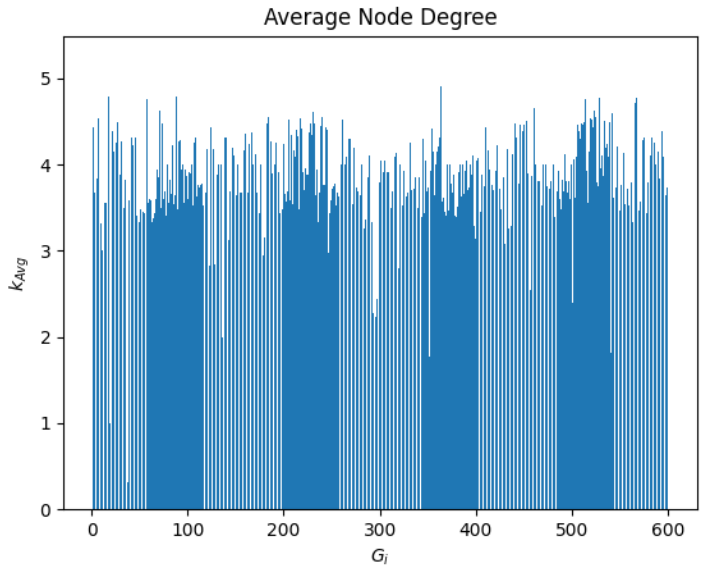
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



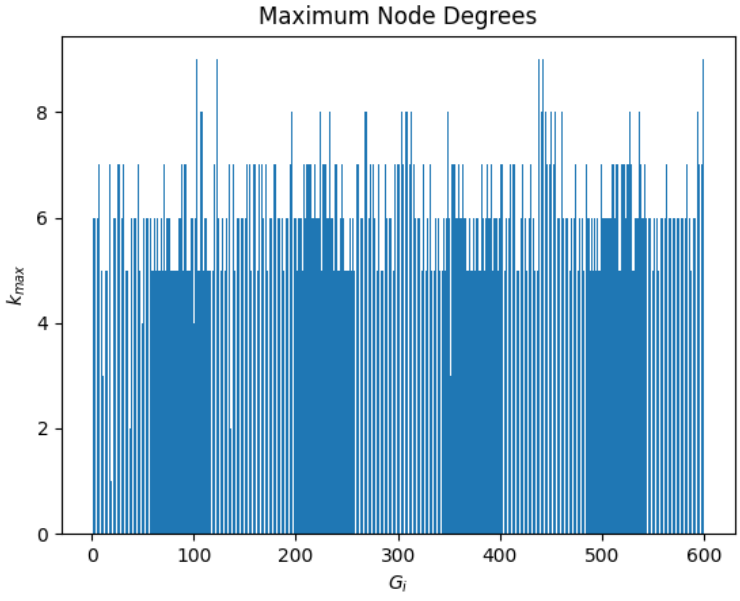
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



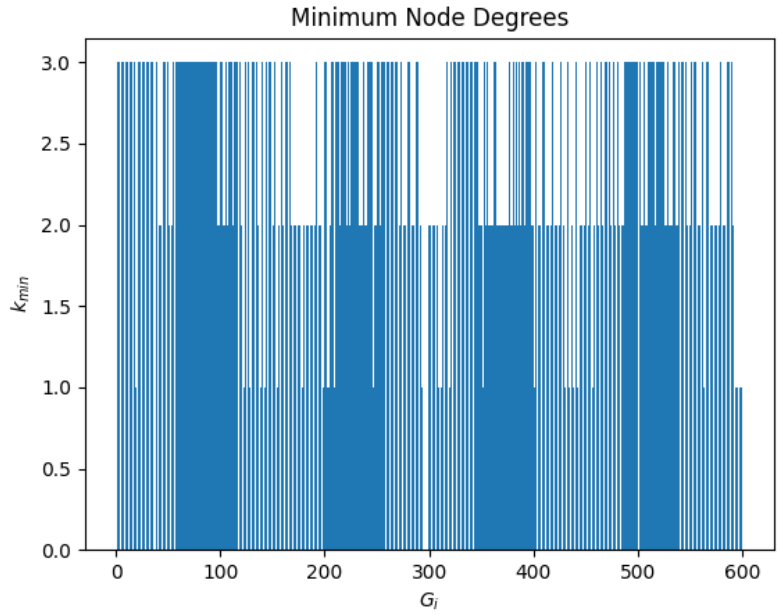
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



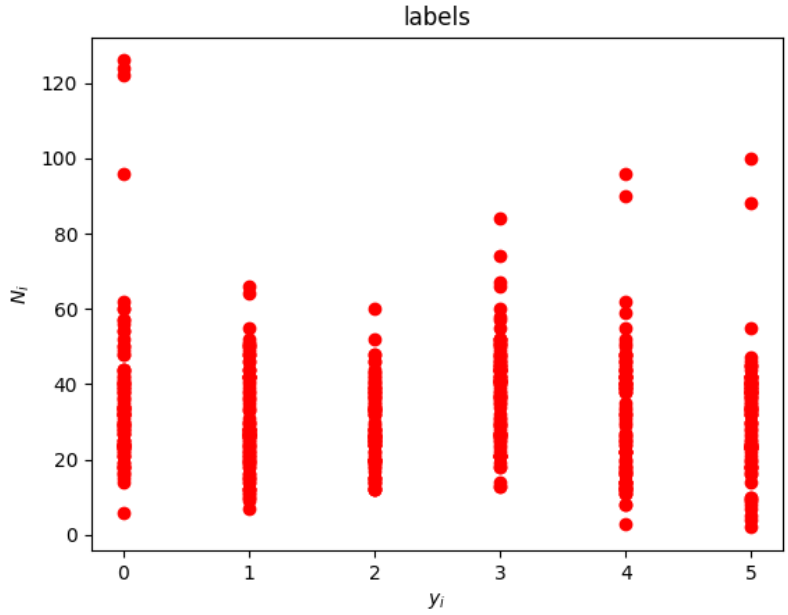
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



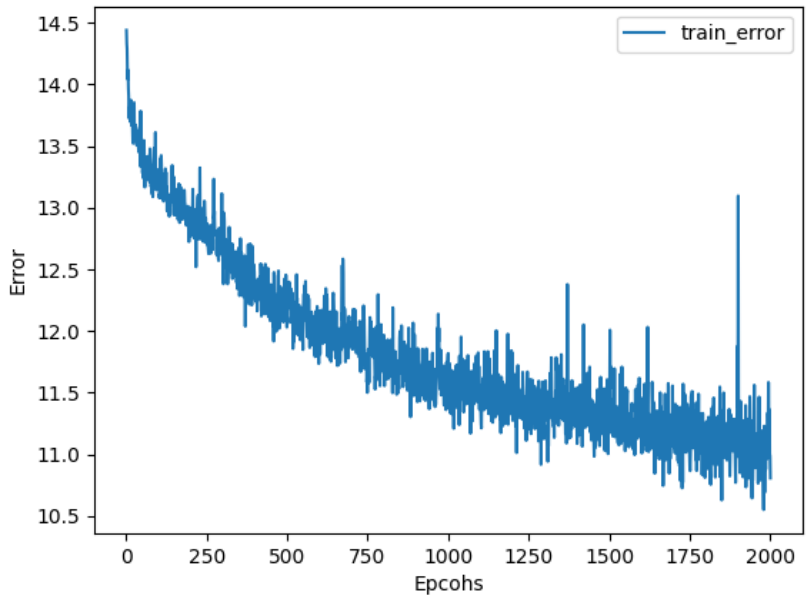
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



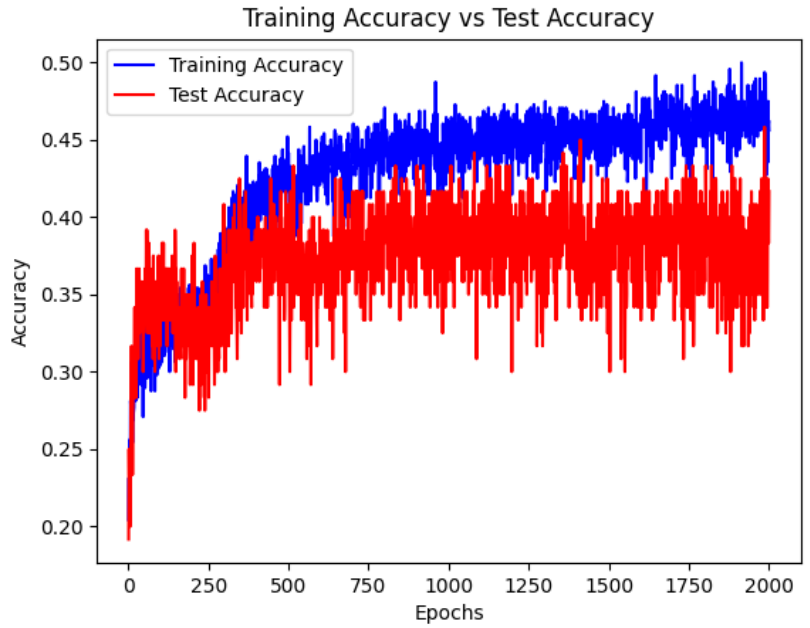
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



6.7 MCF-7 Dataset :

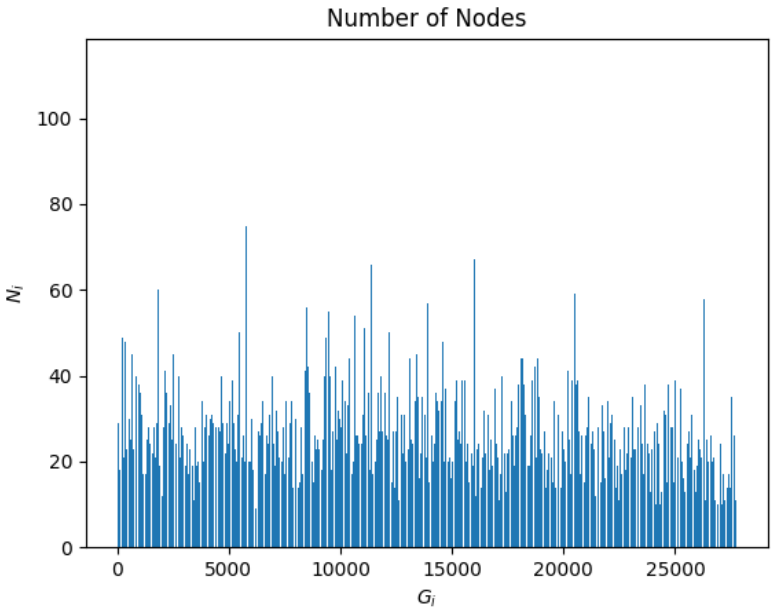
The MCF-7 dataset is a synthetic dataset where its data is generated artificially for research purposes.The MCF-7 dataset cell line which is popularly used in human breast cancer cell line in studies on cancer.It is frequently used to research breast cancer and create machine learning algorithms for classification and predicting tasks.

- Graphs represent Molecular structures(molecule)
- Atoms represented as nodes
- Chemical bonds represent edges

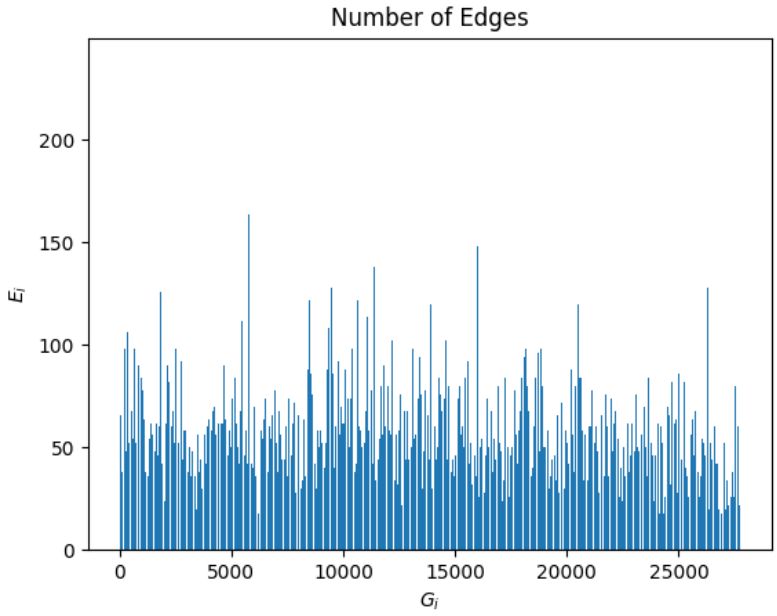
This dataset is typically used for classification tasks, where the goal is to predict a binary or multi-class label for each molecule. The target variable in the dataset represents the label of each molecule, such as whether it is cancerous or not.

6.7.1 Analysis:

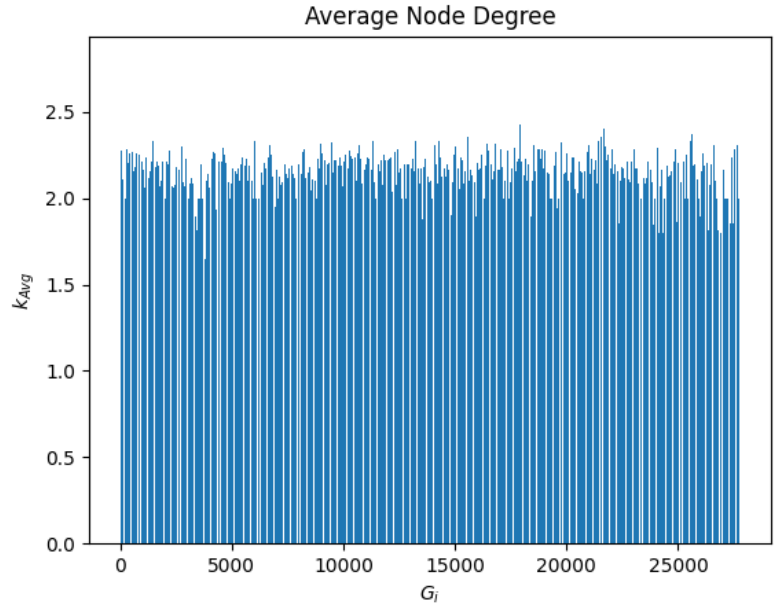
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



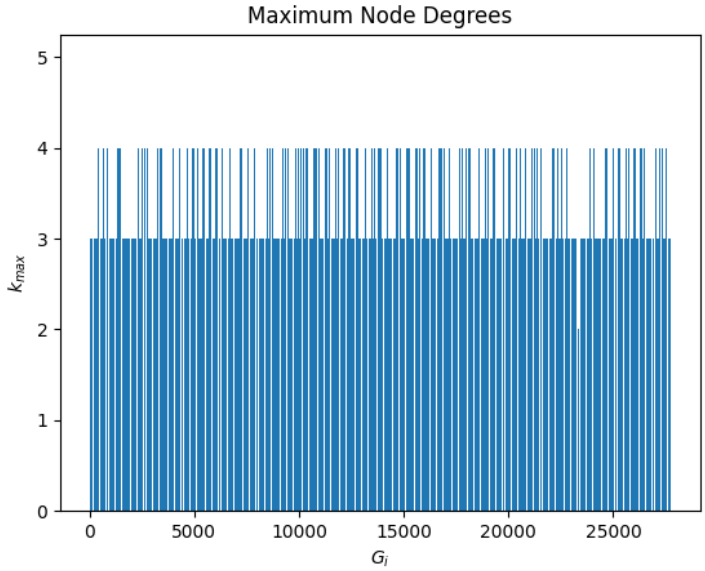
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



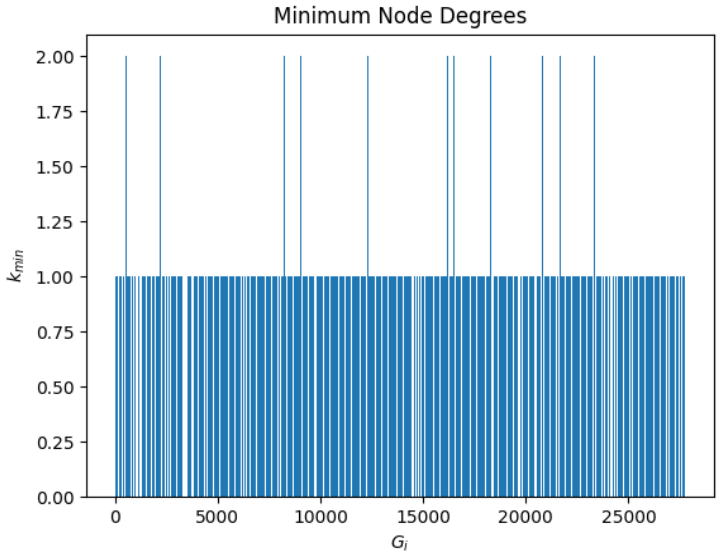
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



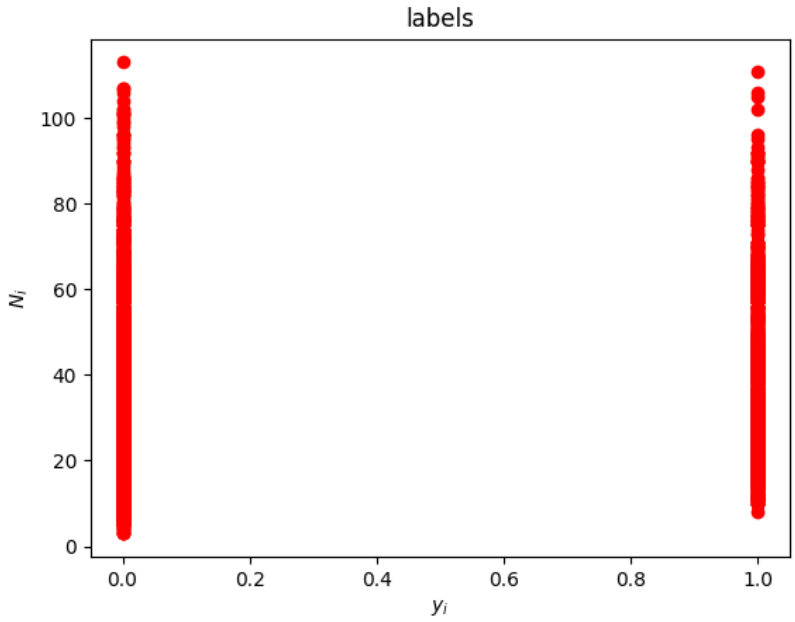
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



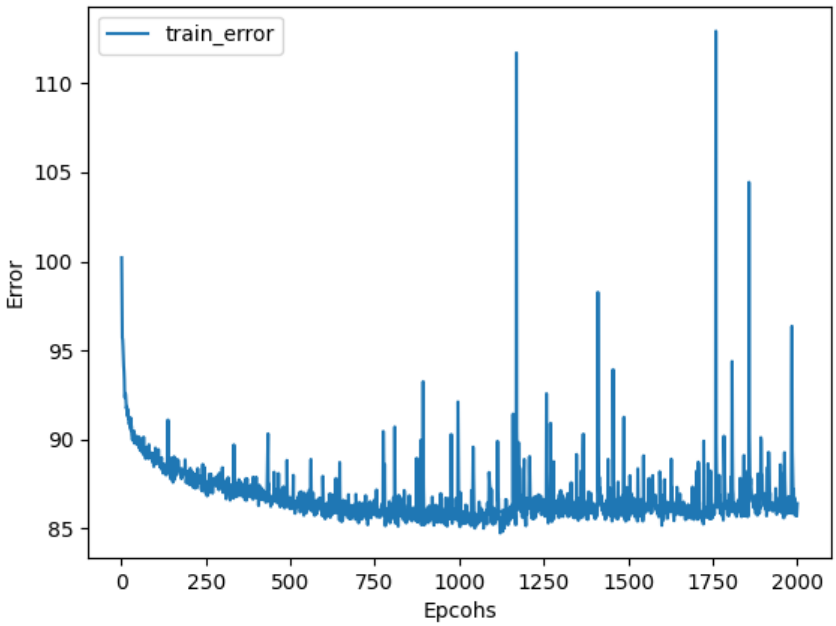
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



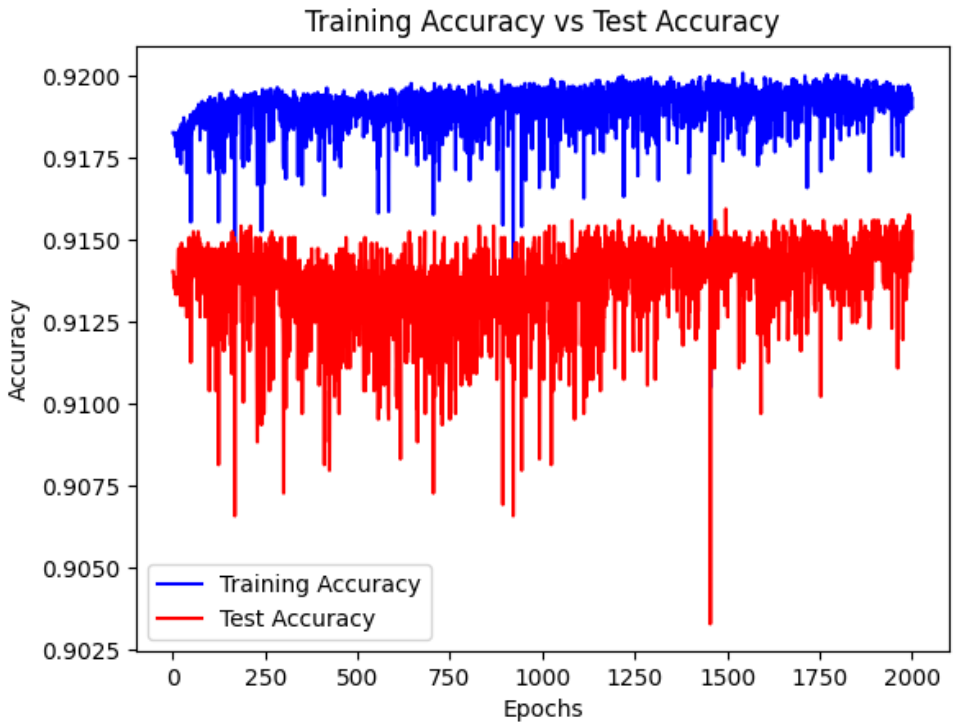
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



6.8 YEAST Dataset :

The YEAST dataset is a protein-protein interaction (PPI) network of the Saccharomyces cerevisiae yeast species. It contains information about protein nodes and their interactions.

- Graphs(79 601) represent Molecule
- Nodes represent proteins
- Edges represent interactions between them
- 0 and 1 tell whether the molecule is yeast or not respectively.

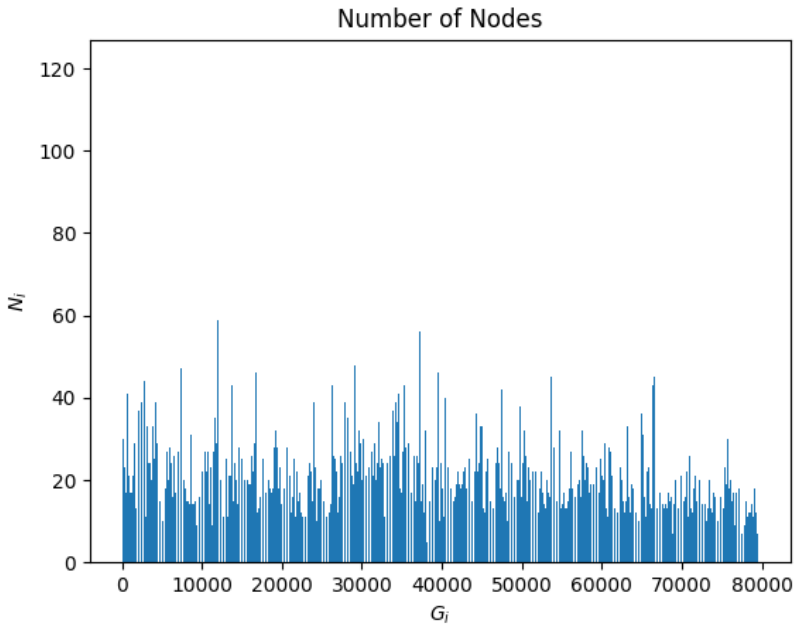
The dataset is derived from the Yeast Interactome Database (YIDB) and has been used in many studies related to protein function prediction, protein-protein interaction prediction, and graph representation learning.

The YEAST dataset does not come with graph labels, as it is an unlabeled dataset.Here researchers generated them separately and then combined them with the YEAST dataset.

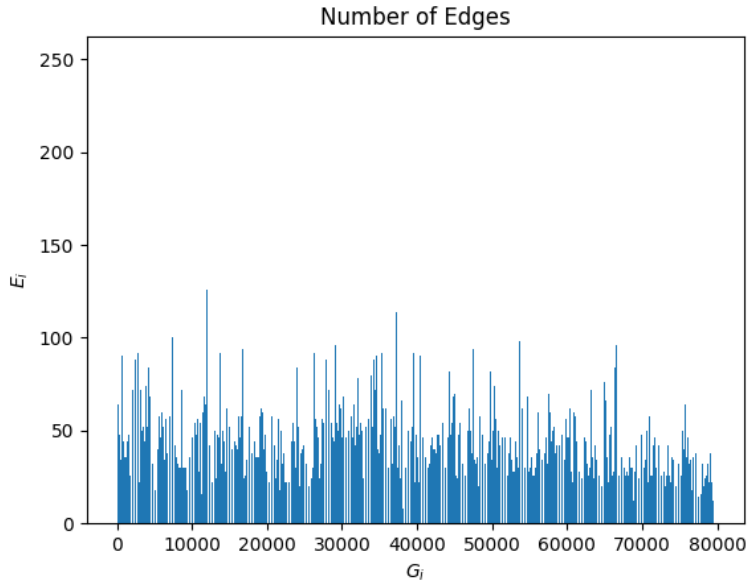
Here they obtain graph labels from external sources and use some domain-specific methods to generate graph-level labels based on the protein nodes and their interactions in the PPI network. These graph labels can then be used for tasks such as graph classification or graph regression, depending on your specific use case.

6.8.1 Analysis:

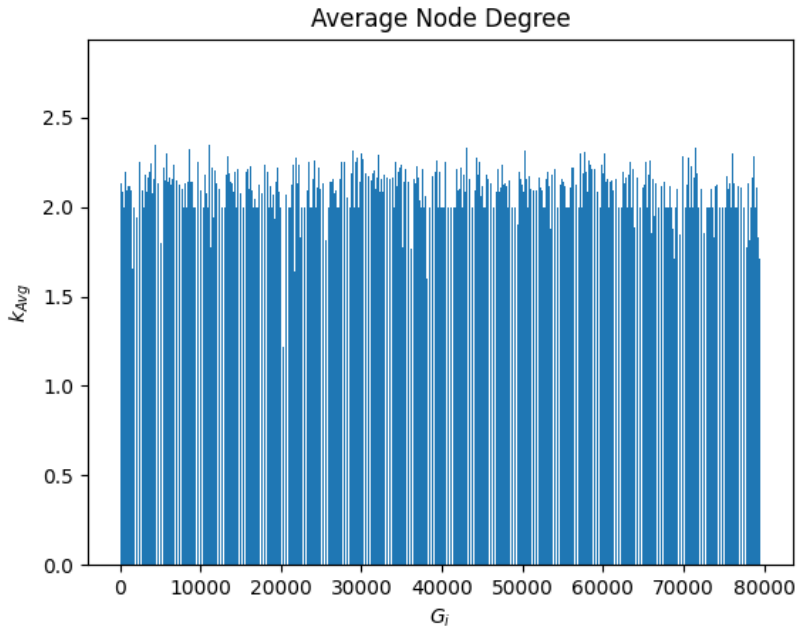
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



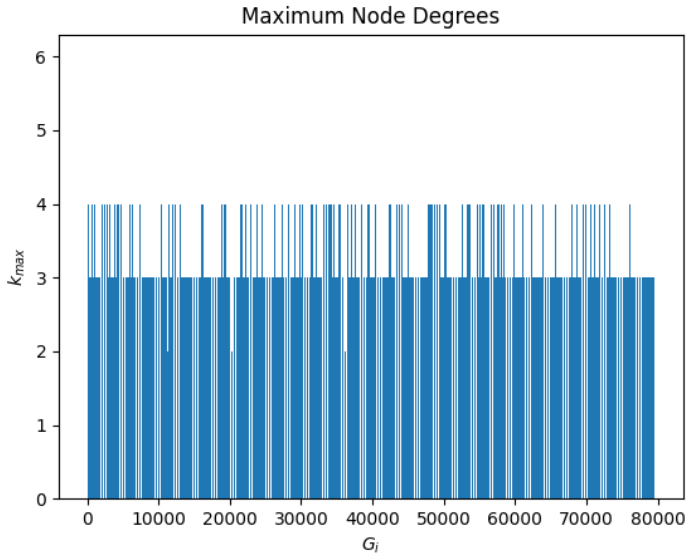
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



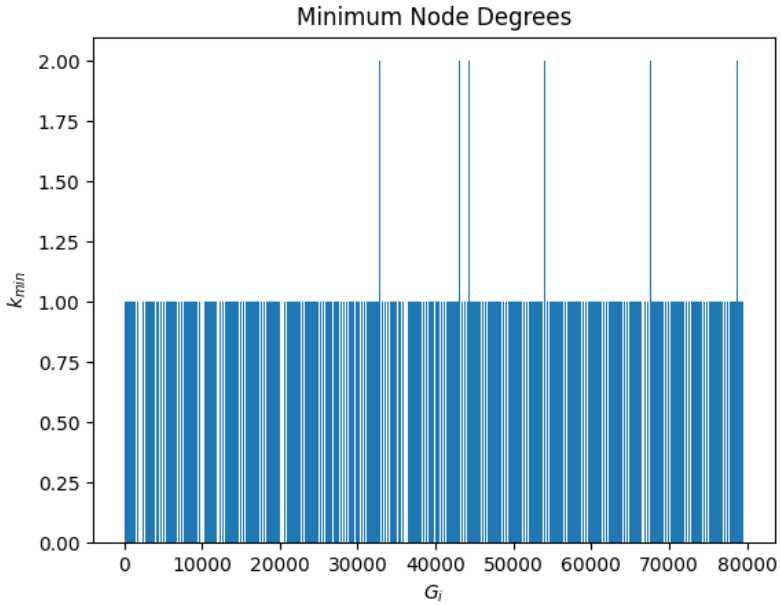
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



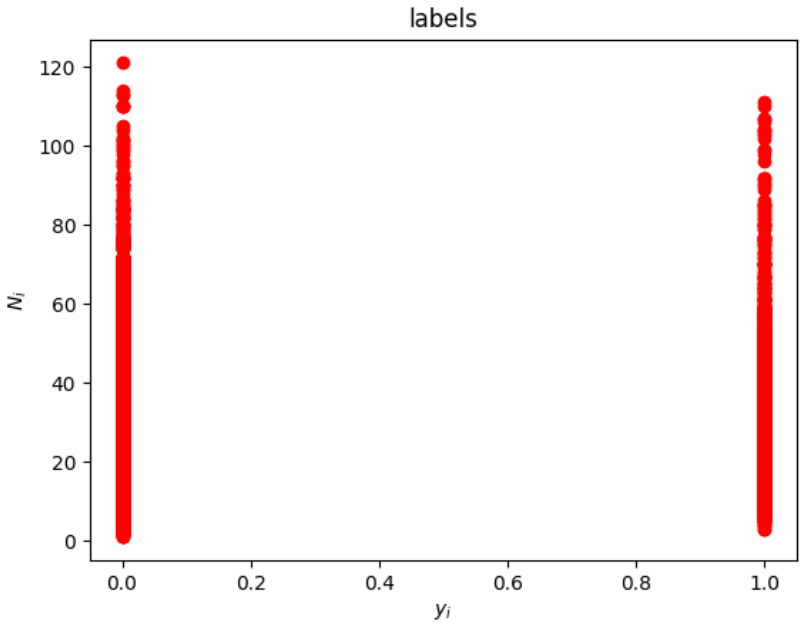
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



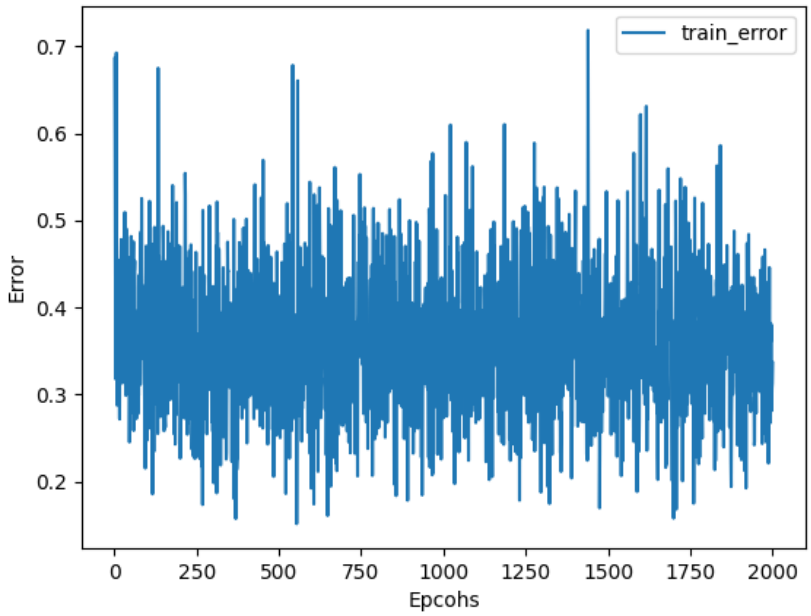
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



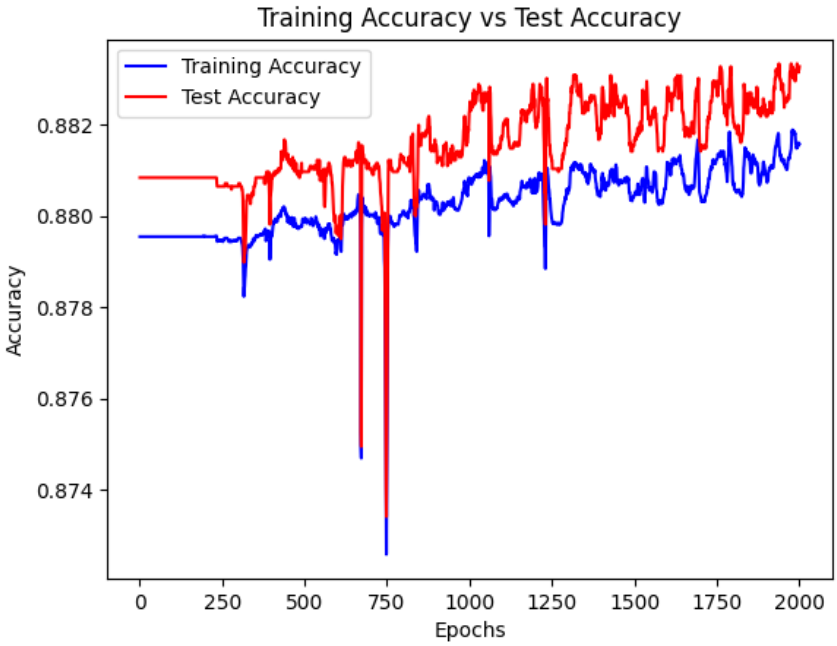
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



6.9 PCT_FM Dataset :

The PTC_FM dataset is a collection of 349 graphs, which is a popular library for deep learning on graph-structured data and it contains graphs that represent protein contact maps derived from Free Modeling (FM) experiments. Contact maps are used in computational protein modeling to represent the spatial proximity between pairs of amino acids in a protein structure.

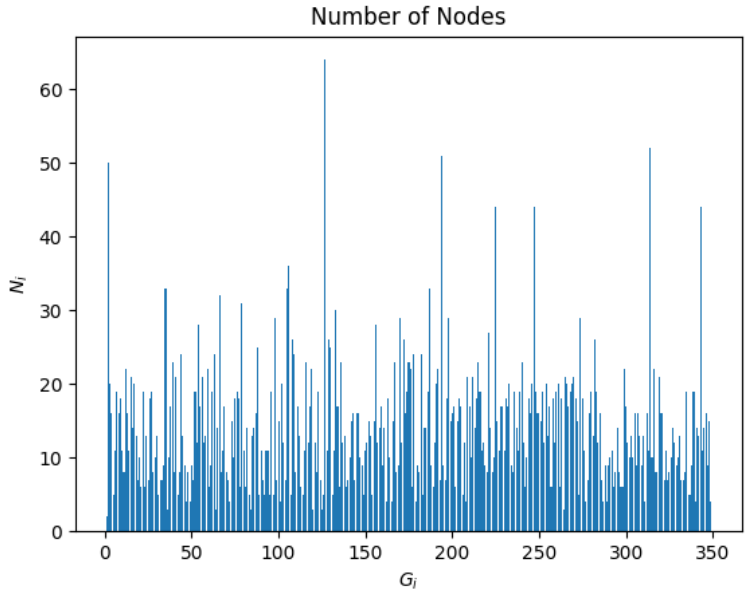
- Graph represents a protein contact map
- Nodes represent amino acids
- Edges Represent contact between pair of amino acids

The PTC_FM dataset is used for tasks such as contact prediction, protein folding prediction, or protein structure prediction,The goal is to predict the three-dimensional structure of proteins.

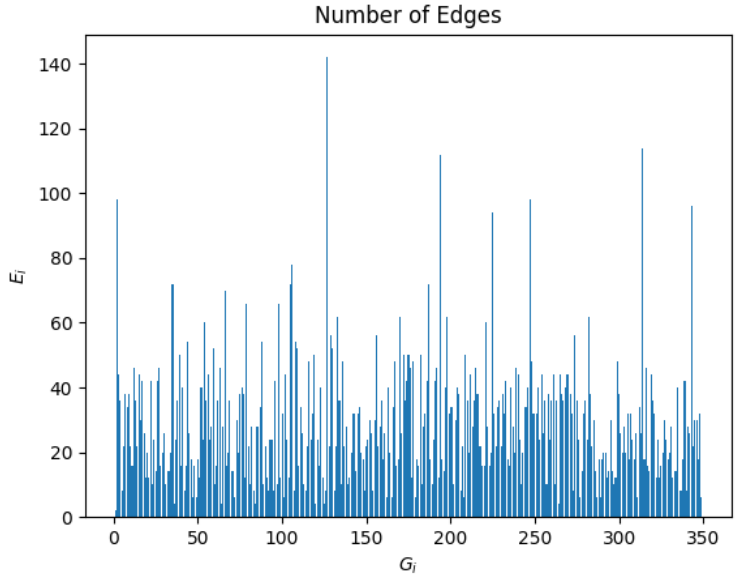
If the structure is correct then 1 or then 0.

6.9.1 Analysis:

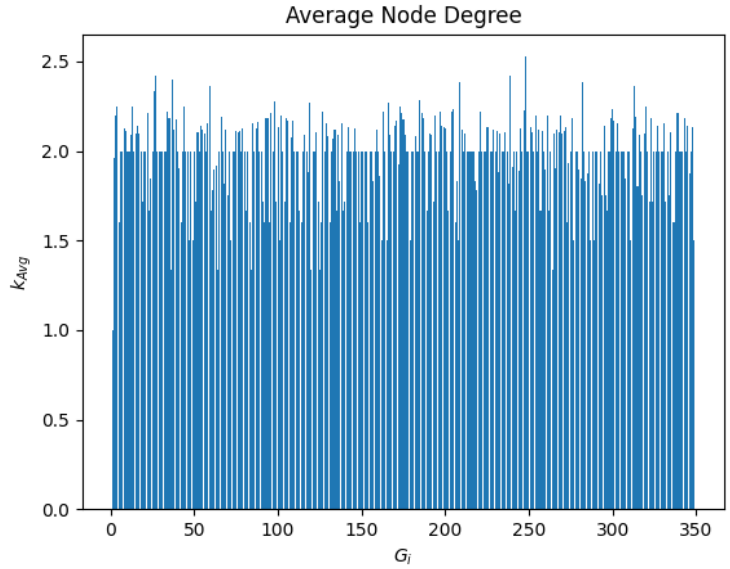
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



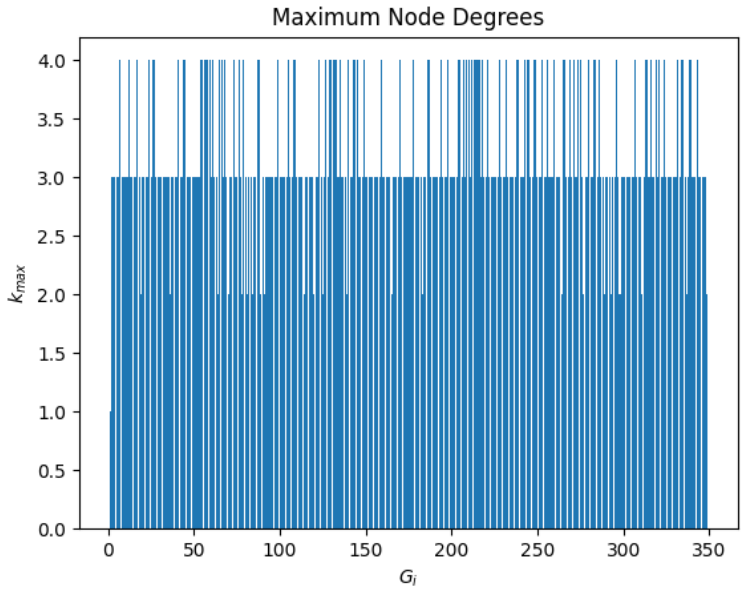
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



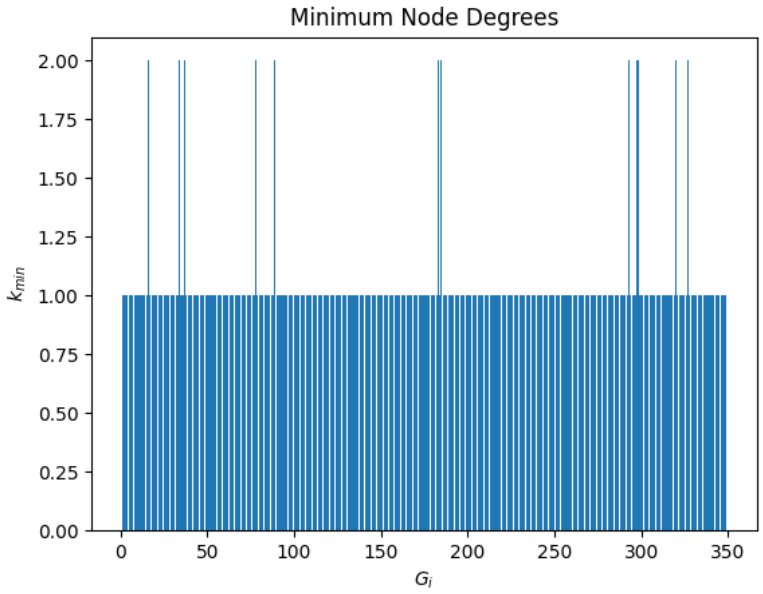
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



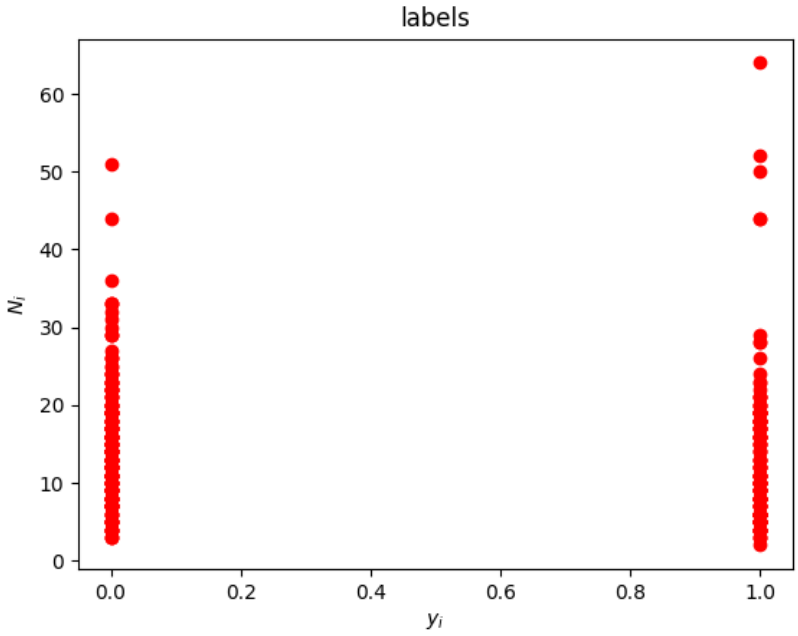
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



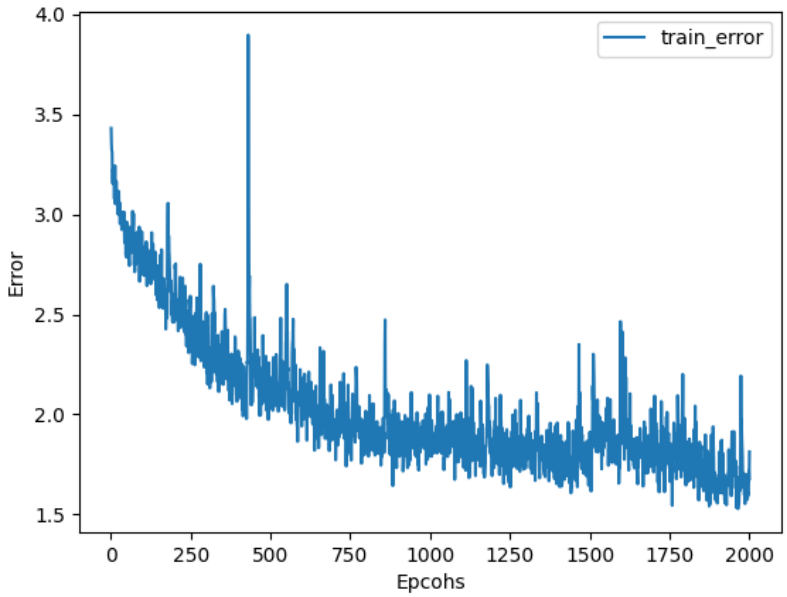
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



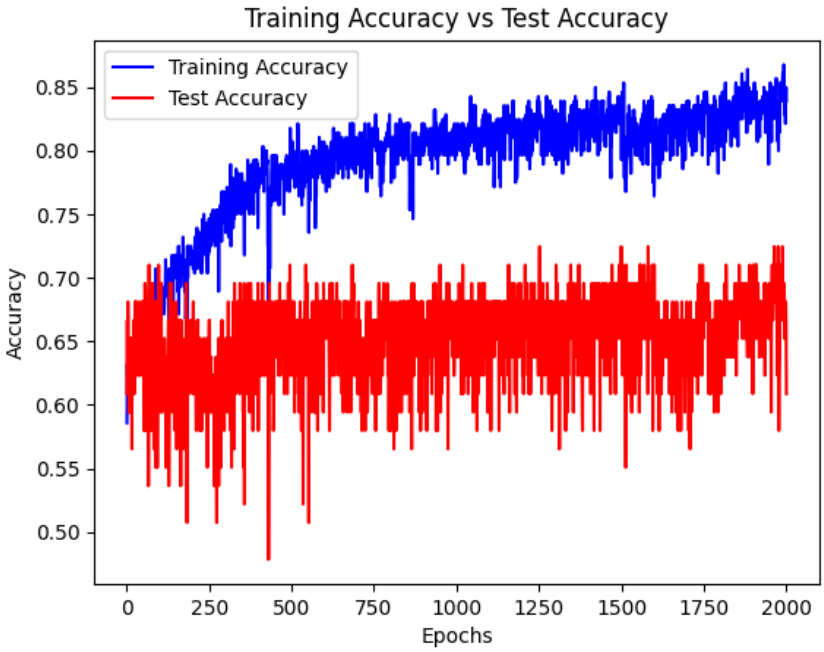
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



6.10 NCI109 Dataset :

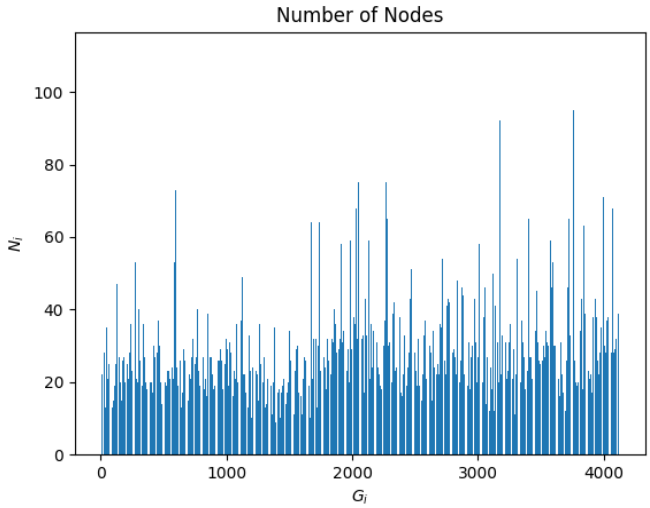
The NCI109 dataset is a collection of 4127 graphs which are used for graph-based machine learning and it is derived from the National Cancer Institute (NCI) Anticancer Screen, which is a publicly available database of biological activity of compounds screened against various cancer cell lines. The NCI109 dataset has been pre processed and transformed into a graph-based format suitable for machine learning tasks.

- Graphs represent Chemical compound
- Nodes represent atoms
- Edges represent chemical bonds

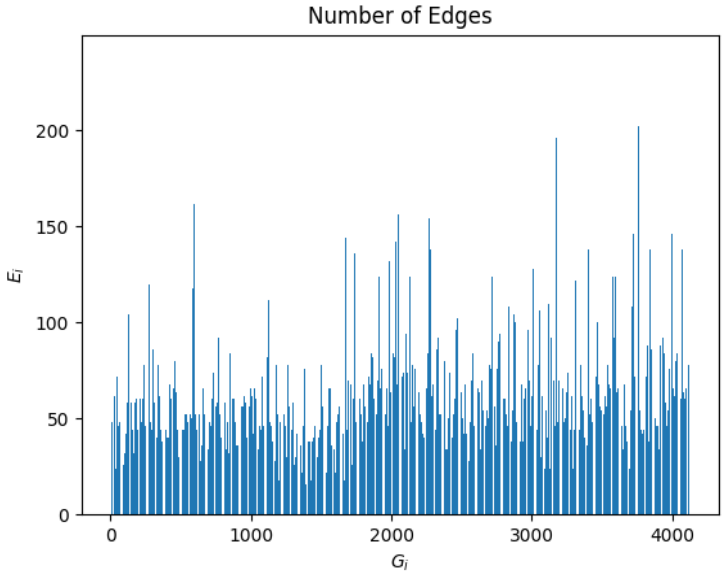
The graphs present in this dataset are fully connected such that there is an edge between every pair of nodes in the graph and the class distribution is usually imbalanced, with a small number of active compounds compared to inactive compounds.Each compound is labeled as either active (1) or inactive (0) based on its biological activity against cancer cell lines.

6.10.1 Analysis:

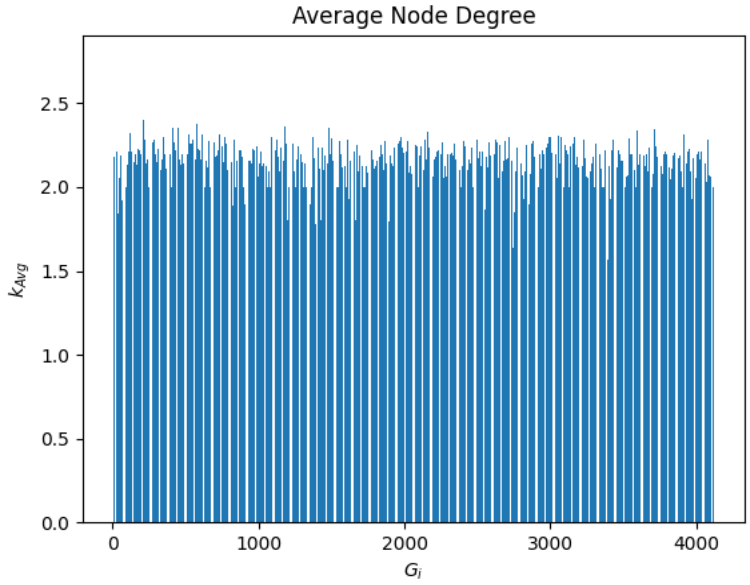
- Plotting Number of nodes of each graph gives us an idea of the range of node sizes that exist in the graphs and how frequently they occur.



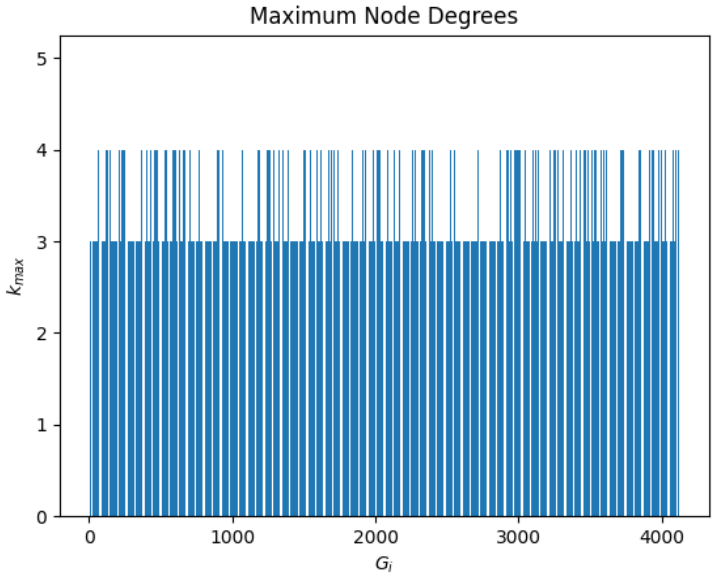
- Plotting Number of edges for each graph tells us how edges for each graph are assigned and distribution of edge density in the set of graphs.



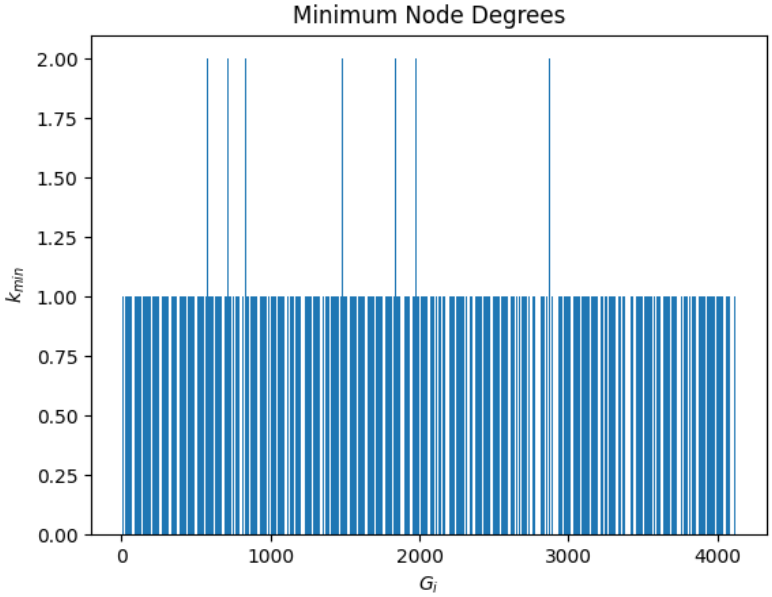
- Plotting the Average Node degree for each graph tells us the increasing or decreasing trend in the average node degree and how edges are distributed among each node in the graph.



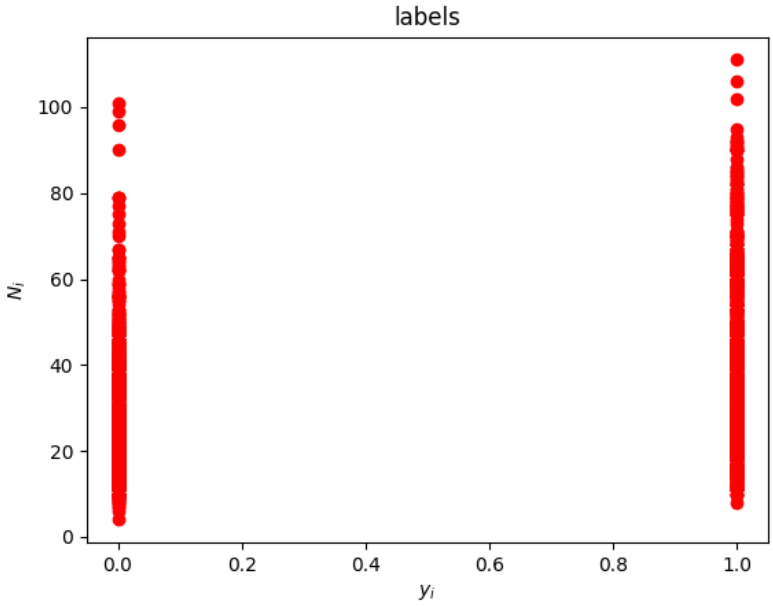
- Plotting a Maximum node degree for each graph we can see the range of maximum node degrees that occur in the set of graphs and how frequently each degree occurs.



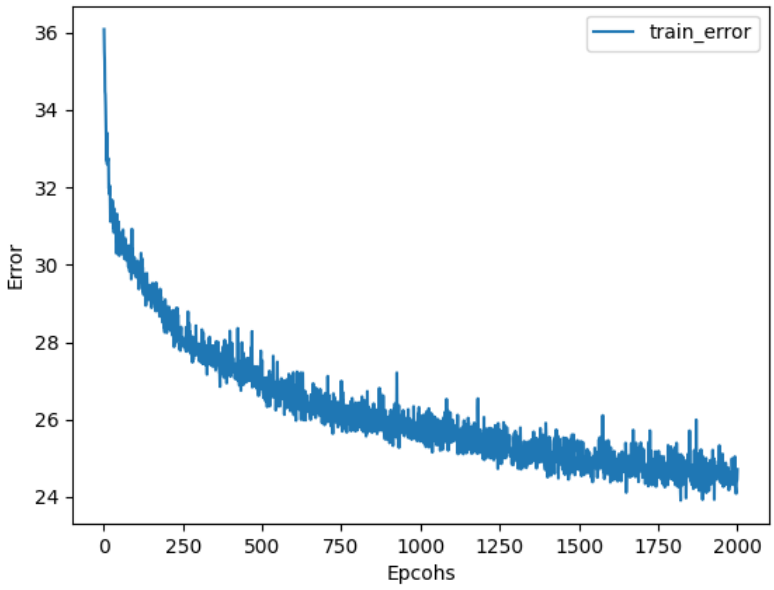
- Plotting a Minimum node degree for each graph we can see the range of minimum node degrees that occur in the set of graphs and how frequently each degree occurs.



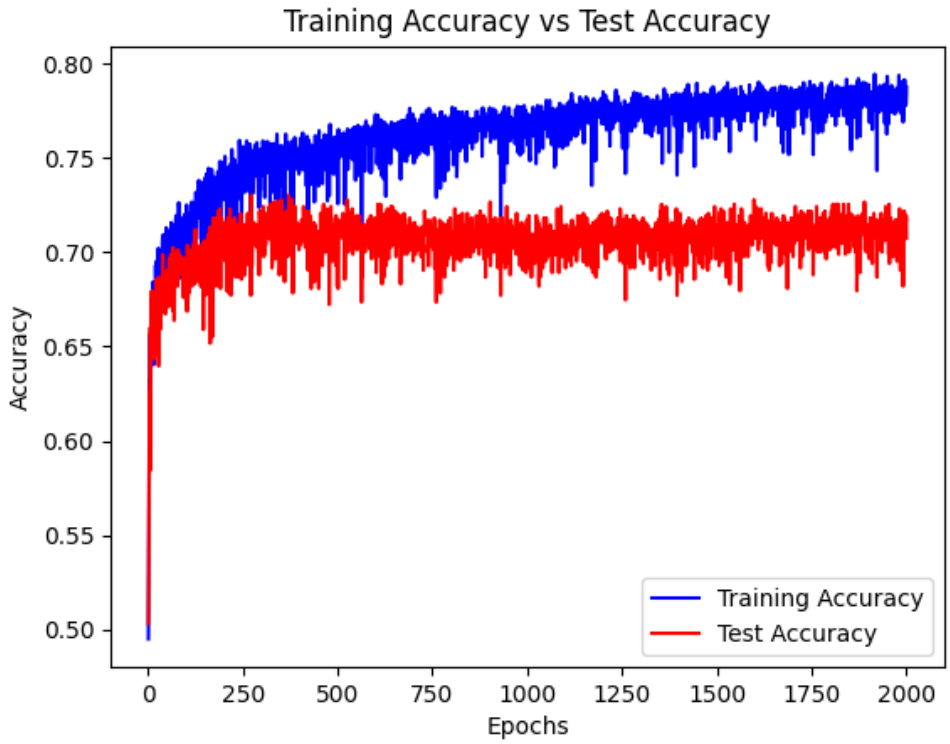
- Plotting a Label for each graph tells us how labels are assigned to graphs of different sizes.



- Plotting a Loss for each graph tells us how loss is decreasing over time as the model learns to make better predictions on the training data.



- Plotting a Accuracy for each graph tells us how model accuracy increasing as the model continues to learn from the training data over every epoch



7. Future work and Conclusion

Feature work : We also performed graph classification on localized and delocalized networks using binary class labels and understood the properties and linear dynamics. We will apply multi class labels on localized and delocalized networks and understand the Properties.

Conclusion : The research proposes a method for constructing localized and delocalized graph neural networks (GNNs) by using the graph Normalization as a basis for feature extraction.we demonstrate the effectiveness of this approach in solving a graph classification problems on degree heterogeneity and degree homogeneity using a variety of datasets of molecular graphs,barabasi albert graphs, erdos renyi graphs and some other graphs like wheel graph, star graph, path graph etc...

This method shows that localized and delocalized networks function more accurately and efficiently than traditional GNNs which has the potential to improve the performance of GNNs in a variety of applications and reduce the overfitting, poor generalization, and difficulty in detecting low-degree nodes.

8.Bibliography

[1] J. T. Vogelstein, W. Gray Roncal, R. J. Vogelstein and C. E. Priebe, "Graph Classification Using Signal-Subgraphs: Applications in Statistical Connectomics," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 7, pp. 1539-1551, July 2013, doi: 10.1109/TPAMI.2012.235.

[2] Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.

[3] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29.

[4] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[5] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017, July). Neural message passing for quantum chemistry. In the International conference on machine learning (pp. 1263-1272). PMLR.

[6] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30..

[7] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018, July). Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 974-983).

[8] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637.

[9] Seenappa, M. G. (2019). Graph Classification using Machine Learning Algorithms.

[10] Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018, April). An end-to-end deep learning architecture for graph classification. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

[11] Mueller, T. T., Paetzold, J. C., Prabhakar, C., Usynin, D., Rueckert, D., & Kaissis, G. (2022). Differentially private graph classification with gnns. arXiv preprint arXiv:2202.02575.

[12] Tixier, A. J. P., Nikolentzos, G., Meladianos, P., & Vazirgiannis, M. (2019). Graph classification with 2d convolutional neural networks. In Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28 (pp. 578-593). Springer International Publishing.

[13] Ma, T., Wang, H., Zhang, L., Tian, Y., & Al-Nabhan, N. (2021). Graph classification based on structural features of significant nodes and spatial convolutional neural networks. Neurocomputing, 423, 639-650.

[14] de Lara, Nathan, and Edouard Pineau. "A simple baseline algorithm for graph classification." arXiv preprint arXiv:1810.09155 (2018).

[15] Errica, F., Podda, M., Bacciu, D., & Micheli, A. (2019). A fair comparison of graph neural networks for graph classification. arXiv preprint arXiv:1912.09893.

[16] Pham, T., Tran, T., Dam, H., & Venkatesh, S. (2017). Graph classification via deep learning with virtual nodes. arXiv preprint arXiv:1708.04357.

[17] Karagiannakos, Sergios. 'Best Graph Neural Networks Architectures: GCN, GAT, MPNN and More'. Https://Theaisummer. Com/, 2021, https://theaisummer.com/gnn-architectures/.

[18] P. Pradhan, A. Yadav, S. K. Dwivedi, and S. Jalan, "An Optimized evolution of networks for principal eigenvector localization," Phys. Rev. E,vol. 96, 2017, Art. no. 022312.

[19] D. K. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," pp. 2224–2232, 2015.

[20] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2016. [Online]. Available: