

## Chapter 2 Foundations of Deep Learning

Deep learning is a subfield of machine learning that leverages neural networks to model and solve complex problems. These neural networks are inspired by the structure and function of the human brain. A biological neuron consists of a cell body, dendrites, and an axon (Ref: Figure 2.1). Dendrites receive signals from other neurons, which are processed in the cell body. Axons transmit electrical impulses to communicate information throughout the nervous system.

Similarly, artificial neural networks consist of interconnected nodes known as neurons, organized into layers. These artificial neurons, often referred to as perceptrons, receive input signals, process them, and produce an output. This analogy between biological and artificial neurons serves as the foundation of neural network architecture, allowing for the modeling and solution of complex problems.

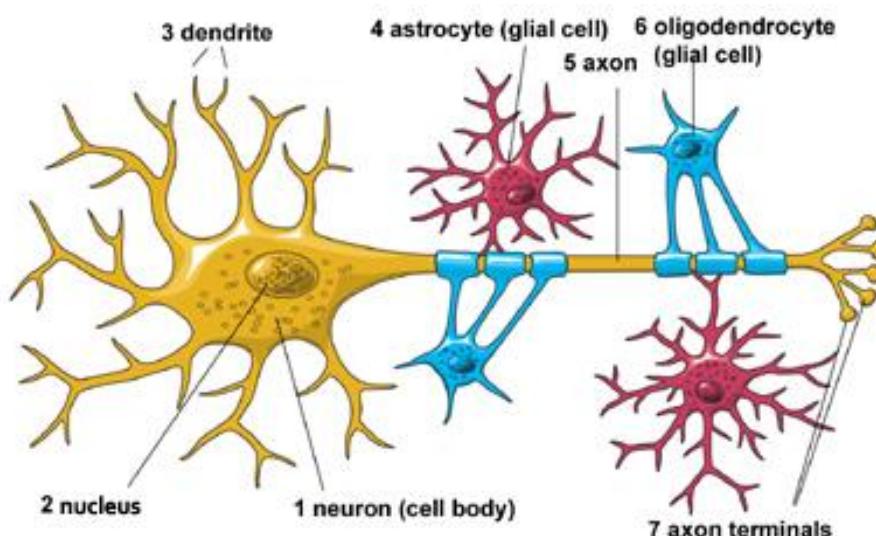


Figure 2.1. Neuron Architecture (Image taken from National Institute of Neurological Disorders and Stroke, United States of America, accessed on 20-May-2024).

Deep Learning Model များသည် လူသားများ၏ အာရုံကြောစနစ်၌ neuron များ၏ အလုပ်လုပ်ပုံကို နှမူနာယူ၍ တည်ဆောက်ထားခြင်းဖြစ်သည်။ neuron များသည် အချက်အလက် များကို ပို့ဆောင် ပေးခြင်း၊ သယ်ယူ ပေးခြင်းတာဝန်ကို ထမ်းဆောင် သည့် messenger များဖြစ်ကြသည်။

neuron တစ်ခုတွင် ပြင်ပ (သို့မဟုတ်) အခြား neuron များမှ အချက်အလက် (signal) များကို ရယူမည့် wire မျှင်များ (dendrites)၊ ပြင်ပသို့ အချက်အလက်များ ပေးပို့မည့် Axon ဟု ခေါ်သည့် output wire များ နှင့် neuron ၏ အမိက ခန္ဓာကိုယ် (cell body) ဟု အပိုင်း (၃) ပိုင်းပါဝင်သည် (Ref: Figure 2.1)။ dendrites မှတဆင့် ရလာသည့် အချက်အလက်များကို neuron ၏ အမိက ခန္ဓာကိုယ် (cell body)မှ တွက်ချက်မှုများ ပြုလုပ်ပြီးနောက် ရလာသည့် ရလဒ်များကို Axon ဟု ခေါ်သည့် output wire များ မှ တဆင့် အခြား neuron များသို့ ပို့ပေးပါသည်။ထိုသို့ပေးပို့လာသည့် အချက်အလက် (signal) များကို အခြား neuron တစ်ခုမှ လက်ခံရယူပြီး အချက်အလက်များ ဆက်လက်ပို့ဆောင်ခြင်းများ ပြုလုပ်ကြပါသည်။ ထိုကဲ့သို့ neuron များ ချိတ်ဆက် အလုပ်လုပ်ခြင်းကို neural network ဟု ခေါ်ဆိုခြင်း ဖြစ်သည်။

လူသားများ၏ အာရုံကြောစနစ်ကို အခြေခံ၍ Artificial neural network architecture များကို Artificial neurons များစွာ ပါဝင်သည့် layer များဖြင့် ချိတ်ဆက် ဖွဲ့စည်းထားသည်။ Layer တစ်ခုတွင် တစ်ခု ထက်ပို့သော neuron များ ပါဝင်ပြီး အဆိုပါ neuron များမှ အချက်အလက်ရယူခြင်း၊ တွက်ချက်ခြင်းနှင့် ရလဒ်များထုတ်ပေးခြင်းတို့ကို ထမ်းဆောင်ကြသည်။ neural network architecture တစ်ခုတွင် input layer ၊ တစ်ခုထက်ပို့သော hidden layer များ နှင့် ရလဒ်ထုတ်ပေးသည့် output layer (neuron) တို့ ပါဝင်သည်။

## 2.1 Neural Network Architecture

Deep learning models excel at tackling complex problems by leveraging neural networks with multiple hidden layers. However, to lay a solid foundation, we begin with one-layer neural networks before moving onto multi-layer neural networks.

### 2.1.1 Single-Layer Neural Network

The simplest form of a neural network is the one-layer neural network, also known as a single-layer perceptron. This basic building block helps in understanding the more complex architectures used in deep learning. A one-layer neural network consists of three main components:

- ❖ **Input Layer:** Takes input features.
- ❖ **Weights and Bias:** Each input is assigned a weight, and a bias term is added.
- ❖ **Activation Function:** Computes the output of the neuron.

Deep Learning Model များသည် hidden layer များစွာပါဝင်သော neural network architecture များဖြစ်သည်။ သို့သော် neural network ၏ အခြေခံ သဘောတရားကို နားလည်စေရန် အတွက် layer တစ်ခုသာ ပါဝင်သော one-layer neural network မှ စ၍ ရှင်းပြသွားမည် ဖြစ်သည်။ One-layer neural network သည် အလွယ်ကူ အရိုးရှင်းဆုံးသော neural network တစ်ခု ဖြစ်ပြီး အစိတ်အပိုင်း (၃)ခု ပါဝင်သည်။

- ❖ **Input Layer:** Input layer တွင် ပါဝင်သည့် neuron တစ်ခုချင်းစီသည် Input feature တစ်ခုစီနှင့် တိုက်ရှိက်ချိတ်ဆက်ထားသည်။
- ❖ **Weights and Bias:** weight (မြောက်ဖောက်ကိန်း) များသည် Input feature တစ်ခုချင်းစီ၏ အရေးပါမှုကို ဆုံးဖြတ်ပေးသည်။ ထို့အပြင် Bias Unit တစ်ခုကိုလည်း ထည့်သွင်းစဉ်းစားရမည်။
- ❖ **Activation Function:** Activation Function သည် တွက်ချက်မှုများကို ပြုလုပ်ပြီး ရလဒ်ကို ရှာပေးမည့် Function ဖြစ်သည်။

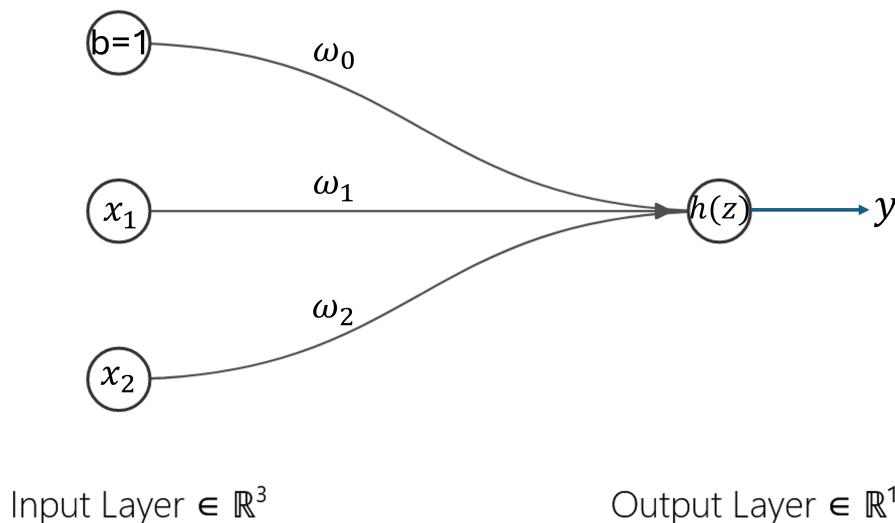


Figure 2.2. Example of Single-Layer Perception (One Layer Neural Network) with two features. Image is generated using NN-SVG[12]

Figure 2.2 illustrates a single-layer neural network with two input features. For example, suppose this network models the prediction of a company's sales amount based on TV and radio advertisement costs. The nodes ( $x_1$ ), ( $x_2$ ), represent these input features, while ( $b = 1$ ) serves as a bias unit. The bias unit ( $b = 1$ ) is a constant added to the input to enhance model fitting. The weights ( $\omega_1$ ) and ( $\omega_2$ ) determine the importance of the associated features. The activation function  $h(z)$  applies a transformation to the weighted sum of inputs and produces the output. During training, these weights are adjusted iteratively to minimize errors and ensure alignment between predicted and actual outputs.

ဥပမာ - အထက်ပါပုံတွင် ပြဿနာသည့် single-layer neural network သည် ကုမ္ပဏီတစ်ခု၏ ရောင်းအားကို ခန့်မှန်းပေးမည့် Machine Learning Model တစ်ခုဟု ဆိုကြပါစိုး။ ကုမ္ပဏီ၏ ရောင်းအားကို TV နှင့် radio ကြော်ပြာခ အတွက် အသုံးပြုသည့် ကုန်ကျစရိတ်များကို မူတည်၍ ခန့်မှန်းမည်။ ဤဥပမာတွင် nodes ( $x_1$ ) နှင့် ( $x_2$ ) သည် TV နှင့် radio ကြော်ပြာခ များကို ကိုယ်စားပြုသည်။

( $b = 1$ ) သည် ကြော်ပြာခ လုံးဝ အသုံးမပြုသည့် အခြေအနေတွင် ဖြစ်ပေါ်နိုင်သည့် ရောင်းအားကို သိရှိနိုင်ရန် ထည့်ပေးသည့် ကိန်းသေတစ်ခုဖြစ်ပြီး bias unit ဟု ခေါ်သည်။ မြောက်ဖော်ကိန်း ( $\omega_1$ ) နှင့် ( $\omega_2$ ) များသည် ရောင်းအားကို တွက်ချက်ရာတွင် TV နှင့် radio ကြော်ပြာခ တို့၏ အရေးပါမှုနှင့် တိုက်ရှိက်အချိုးကျသည်။ ဥပမာ - TV ကြော်ပြာခ တိုးလိုက်သည့် အချိန်တွင် ရရှိသည့် ရောင်းအား ပမာဏသည် radio ကြော်ပြာခ တိုးလိုက်သည့် အချိန်တွင် ရရှိသည့် ရောင်းအားပမာဏ၊ ထက်များနေမည်ဆိုပါက ( $\omega_1$ ) ၏ တန်ဖိုးသည် ( $\omega_2$ ) ၏ တန်ဖိုးထက်များမည် ဖြစ်သည်။

Machine Learning Model တစ်ခုတည်ဆောက်သည် (သို့မဟုတ်) Train လုပ်သည် ဆိုသည်မှာ တနည်းအားဖြင့် အထက်ပါ မြောက်ဖော်ကိန်း ( $\omega_1$ ) နှင့် ( $\omega_2$ ) များ၏ တန်ဖိုးကို ရှာဖွေခြင်းပင် ဖြစ်သည်။

## 2.1.2 Multi-layer Neural Network

A multi-layer neural network, specifically a Multi-Layer Perceptron (MLP), extends the concept of the single-layer perceptron by introducing one or more hidden layers between the input and output layers. This additional complexity allows the network to learn and represent more intricate patterns and relationships in the data, making it capable of solving more complex problems.

An MLP typically consists of three types of layers:

- ✿ **Input Layer:** The layer that receives the input features.
- ✿ **Hidden Layers:** One or more intermediate layers where the computation happens. These layers perform transformations on the input data.
- ✿ **Output Layer:** The final layer that produces the output, such as class probabilities in classification tasks.

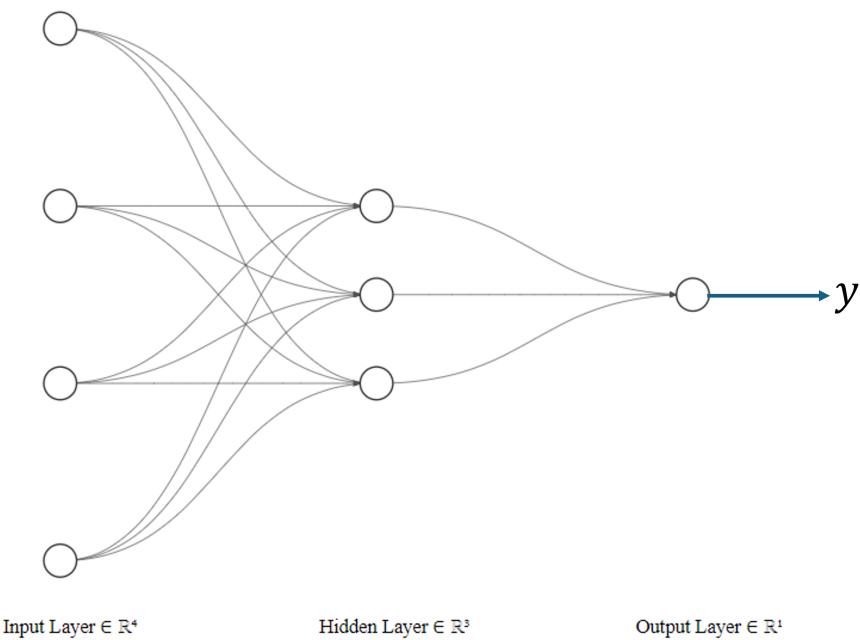


Figure 2.3. Example of Multi-Layer Perception with one hidden layer. Image is generated using NN-SVG[12]

Multi-layer neural network အထူးသဖြင့် Multi-Layer Perceptron သည် single-layer perceptron (သို့) neural network တွင် hidden layer များထည့်သွင်းဖြစ်သည်။ Multi-Layer Perceptron တစ်ခုတွင်

- ✿ **Input Layer:** Input layer တွင် ပါဝင်သည့် neuron တစ်ခုချင်းစီသည် Input feature တစ်ခုစီနှင့် တိုက်ရှိက်ချိတ်ဆက်ထားသည်။
- ✿ **Hidden Layers:** ကြားခဲ့တဲ့ layer များဖြစ်ပြီး ငြင်းလဲတဲ့ layer များသည် Input feature များကို အဆင့်ဆင့် ပြောင်းလဲ ပေးသွားသည်။
- ✿ **Output Layer:** Neural network ၏ နောက်ဆုံး layer ဖြစ်ပြီး output ကို ထုတ်ပေးသည်။

## 2.2 Activation Functions

The activation function  $h(z)$  computes the final output of the neuron based on its pre-activation value. For a single neuron in the neural network layer shown in Figure 2.2, the pre-activation value  $z$  represents the weighted sum of inputs to a neuron, including the bias term.

$$z = \omega_1 x_1 + \omega_2 x_2 + \omega_0, \quad (2.1)$$

where

- $(\omega_1)$  and  $(\omega_2)$  are the weights associated with the connections from input features to the output neuron.
- $(x_1)$ , and  $(x_2)$ , are the input features (e.g., TV and radio advertisement costs).
- $b = 1$  is the bias term associated with the output neuron.

In general, the equation 2.1 is written as:

$$z = \sum_{i=1}^n \omega_i x_i + \omega_0, \quad (2.2)$$

where  $n$  is the number of features.

Common activation functions include Step, Sigmoid, ReLU (Rectified Linear Unit), and Hyperbolic Tangent (tanh) Function. The choice of activation function depends on the specific task and the characteristics of the data. Experimentation with different activation functions is common to find the one that yields the best performance for a given problem.

အထက်ပါပဲ 2.2 တွင် ပြသထားသည့် single-layer neural networkတွင် pre-activation value,  $z$  ၏ တန်ဖိုးသည် မြောက်ဖော်ကိန်းများနှင့် မြောက်ထားသည့် ကြော်ငြာခ - ၂ ခု ၏ ပေါင်းခြင်းနှင့် ညီမြှုပ်နည်း 2.1 ကို ကြည့်ပါ။ ဤဥပမာတွင် input feature ၂ ခုသာပါဝင်ပြီး  $(x_1)$  နှင့်  $(x_2)$  သည် TV နှင့် radio ကြော်ငြာခ များကို ကိုယ်စားပြသည်။ input feature များစွာ ပါဝင်သော ပုံစံအတွက် ဆိုပါက ညီမြှုပ်နည်း 2.2 ကို အသုံးပြုရမည်။ တန်လုံးဆိုသော pre-activation value,  $z$  သည် မြောက်ဖော်ကိန်းများနှင့် မြောက်ထားသည့် input feature များ၏ စုစုပေါင်းတန်ဖိုးဖြစ်သည်။

Activation function အမျိုးမျိုးရှိပြီး လူသုံးများသည့် function များမှာ Step၊ Sigmoid၊ ReLU (Rectified Linear Unit) နှင့် Hyperbolic Tangent (tanh) Function တို့ဖြစ်ကြသည်။ ပုံစံ၏ သဘောတရား နှင့် ဒေတာ အမျိုးအစား များအပေါ် မူတည်၍ Activation function ကို ရွေးချယ်ကြလေ့ရှိသည်။

### 2.2.1 Step Function

This function produces binary output, typically 0 or 1, based on whether the input exceeds a certain threshold,  $\theta$ . It's commonly used for binary classification tasks. The mathematical representation of the step function,  $h_{step}(z)$  is

$$h_{step}(z) = \begin{cases} 1, & \text{if } z \geq \theta \\ 0, & \text{otherwise} \end{cases} \quad (2.3)$$

where  $\theta$  is the threshold value.

Figure 2.4 illustrates a step function. As can be seen, the x-axis represents the pre-activation value  $z$ , and the y-axis represents the output of the function. It produces a binary value of 0 for all inputs below the threshold and 1 for all inputs above the threshold. The step activation function was historically used in early neural networks. However, it is rarely used in modern deep learning due to its binary nature.

Step Function ကို Binary classification ပုံစံများအတွက် အဓိက အသုံးပြုခဲ့ကြသည်။ Binary classification ပုံစံ နမော အချို့မှာ - စာမေးပွဲ အအောင်၊ အရှုံးသတ်မှတ်ခြင်း၊ ပန်းသီးတစ်လုံး အပုတ်၊ အကောင်းဆုံးဖြတ်ခြင်း စသည် တို့ ဖြစ်သည်။ ရလဒ်မှာ 'အောင်သည်' (သို့မဟုတ်) 'ကျသည်' - အဖြေ ၂ ခု အနက် - တစ်ခုသာ ဖြစ်သည်။ ပဲ 2.4 တွင် Step Function ၏ သဘောတရား

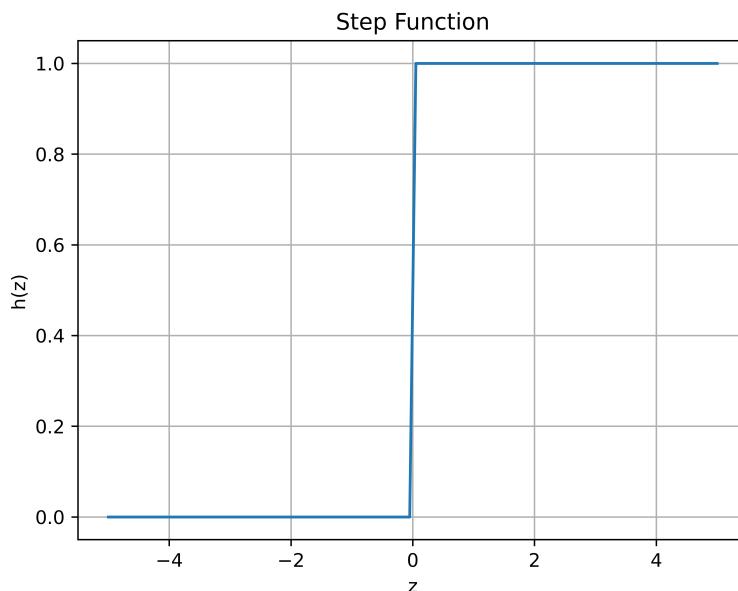


Figure 2.4. Illustration of Step Function. Image is generated using Python.

ကိုဖော်ပြထားသည်။ အကယ်၍ pre-activation value  $z$  ၏ တန်ဖိုးသည် ကြိုက်တင်သတ်မှတ်ထားသည့် ကိန်းသေ (threshold) ထက် နည်းပါက ရလဒ် - သူည် ('ကျသည်') ကို ထုတ်ပေးပြီး ကိန်းသေ (threshold) ထက် များပါက ရလဒ် - တစ် ('အောင်သည်') ဟု ထုတ်ပေးမည် ဖြစ်သည်။

Step Function ကို ယနေ့ခတ်တွင် မသုံးသလောက် နည်းပါးသွားပြီဖြစ်သည်။ ဤ Function ၏ ပြဿနာမှာ ကိန်းသေ (threshold) ၏ နေရာတွင် သူည်မှ တစ်သို့ ရှုတ်တရက် ပြောင်းလဲ သွားခြင်း ဖြစ်သည်။ ဥပမာ - စာမေးပွဲ အအောင်၊ အရှုံး ပြဿနာတွင် - အောင်မှတ် (threshold) ကို ၄၀ ဟုတ် သတ်မှတ်ထားပါက ၄၀.၁ မှတ် ရသော ကျောင်းသားသည် အောင်သော်လည်း ၃၉.၉ မှတ်ရသော ကျောင်းသားသည် ကျမည် ဖြစ်သည်။

## 2.2.2 Sigmoid Function

The sigmoid function is often used in the output layer for problems where the output needs to be probabilistic or numerical. It squashes the input values to the range between 0 and 1, which can be interpreted as probabilities. For instance, in a binary classification problem, the output of the sigmoid function can represent the probability of belonging to one of the two classes. If the probability is above a certain threshold (e.g., 0.5), the input is classified into one class; otherwise, it's classified into the other class. Mathematically, it is represented as:

$$h_{\text{sigmoid}}(z) = \frac{1}{1 + \exp(-z)} \quad (2.4)$$

where pre-activation value,  $z$  is the input to the function, typically the weighted sum of the inputs plus a bias term.

As shown in Figure 2.5, the sigmoid function is smooth and differentiable, which makes it suitable for gradient-based optimization techniques. However, one drawback is that it can cause the vanishing gradient problem in deep networks, where gradients become very small, slowing down or even preventing the network from training effectively. Despite this, the sigmoid function remains a popular choice for binary classification problems due to its probabilistic interpretation.

Sigmoid Function ကို Neural Network ၏ output layer အတွက် အသုံးပြုသည်။ ပုံ (2.5) တွင် ပြသထားသည့် အတိုင်း Sigmoid Function ၏ output သည် သူည်မှ တစ် အတွင်းရှိပြီး probabilities တန်ဖိုးဟု ယူဆနိုင်သည်။ ဥပမာ - စာမေးပွဲ အအောင်၊ အရှုံး တွက်ချက်သည့် ပုံစံအတွက် Sigmoid Function သည် ကျောင်းသား၏ အောင်နှင့် ချေ ရာခိုင်နှုန်းကို တွက်ချက်ပေးသည်။ အကယ်၍ ကျောင်းသား (သို့မဟုတ်) ကျောင်းသူ ၏ pre-activation value  $z$  သည် သူည် သူည်ထက်နည်းပါက Sigmoid Function ၏ တန်ဖိုးသည် ၀.၅ (သို့မဟုတ်) ၅၀ ရာခိုင်နှုန်းအောက် ဖြစ်မည် (ညီမြှုပြင်း - 2.4 ကို ကြည့်ပါ)။

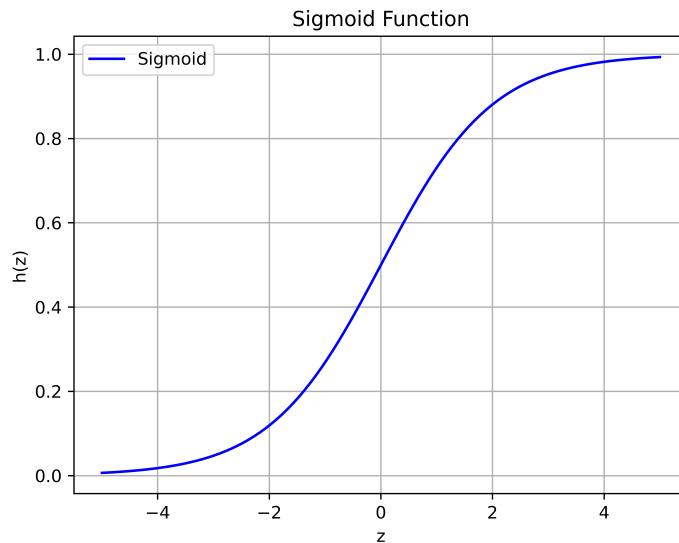


Figure 2.5. Illustration of Sigmoid Function. Image is generated using Python.

ပုံ (2.5) တွင် မြင်ရသည့် အတိုင်း sigmoid function မှပေးသောရလဒ်သည် Step Function ကဲ့သို့ ရှတ်တရက် ပြောင်းလဲသွားခြင်းမျိုး မဟုတ်ဘဲ ဖြည့်ဖြည်းချင်း ပြောင်းလဲ သွားသည်။ pre-activation value  $z$  ၏ တန်ဖိုအနည်း အများပေါ်မှုတည်၍ sigmoid function ၏ ရလဒ်သည် သုည် (သို့မဟုတ်) တစ် သို့ ဖြည့်ဖြည်းချင်း ချုပ်းကပ်သွားသည်။ အဆိုပါ အားသာချက်ကြောင့် sigmoid function ကို Binary classification ပုံစံများအတွက် ယနေ့တိုင် အသုံးပြုနေခဲ့ဖြစ်သည်။

### 2.2.3 Softmax Function

The Softmax function is an activation function commonly used in the output layer of neural networks for multi-class classification tasks. It transforms a vector of raw scores (logits) into a probability distribution, where each class's probability is proportional to the exponential of the logit for that class. The Softmax function ensures that the sum of the probabilities for all classes equals 1, making it possible to interpret the outputs as probabilities. The function is defined as:

$$h_{\text{softmax}}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^K \exp(z_j)} \quad (2.5)$$

where  $z_i$  are the input logits for each class,  $i$  and  $K$  is the number of classes.

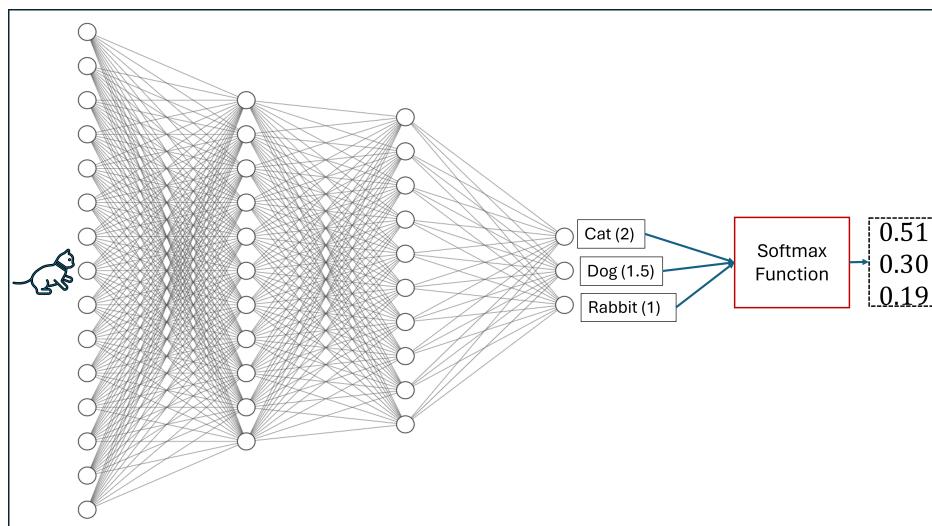


Figure 2.6. Illustration of Softmax Function for Multi-class Classification example. The neural network image was generated using NN-SVG [12].

Figure 2.6 shows an example of a neural network designed to classify images of animals into three categories: cats, dogs, and rabbits. When an image is fed into the neural network, the output layer provides raw scores (logits) for each class. Suppose the logits for a given image are 2 for cats, 1.5 for dogs, and 1 for rabbits. After applying the Softmax function (see Equation 2.5), the probabilities for each class are 0.51 (51%), 0.30 (30%), and 0.19 (19%) respectively.

These probabilities sum to 1, making it possible to interpret them as the likelihood of the image belonging to each class. In this example, the neural network predicts that the image is most likely a cat with a 51% probability.

Softmax function ကို Multi-Class Classification ပုစ္စာများအတွက် တည်ဆောက် သော Neural Network ၏ output layer တွင် အသုံးပြုသည်။ ငြင်းfunction သည် output layer မှ ရရှိသော အကြမ်း ရလဒ်များကို probability တန်ဖိုးများ အဖြစ် ပြောင်းပေးသည်။

ဥပမာ – ပါတ်ပုံတစ်ပုံကို ကြောင်၊ ခွဲ့နှင့် ယုန် အဖြစ် အမျိုးအစား ခွဲခြားပေးမည့် Neural Network တစ်ခုကို တည်ဆောက်ကြသည် ဆိုပါစို့။ Neural Network ၏ output layer မှ တိရစ္ဆာန် အမျိုးအစား တစ်ခုချင်းစီအတွက် ရလဒ် အမှတ်များ ထုတ်ပေးသည်။ ထိုနောက် ပုံ 2.5 တွင် ပြုသထားသည့် အတိုင်း Softmax function မှ အဆိုပါ ရမှတ်များကို probability တန်ဖိုးများ အဖြစ် ပြောင်းပေးသည်။ ရာခိုင်နှုန်းများ၏ ပေါင်းခြေးမှာ တစ် ဖြစ်ပြီး ယခု ပုစ္စာတွင် ပေးထားသည့် ပါတ်ပုံ သည် ကြောင်ဖြစ်နိုင်ချေ ၅၁ ရာခိုင်နှုန်း၊ ခွဲ့ဖြစ်နိုင်ချေ ၃၀ ရာခိုင်နှုန်းနှင့် ယုန်ဖြစ်နိုင်ချေ ၁၉ ရာခိုင်နှုန်း ရှိသည်။

## 2.2.4 ReLU (Rectified Linear Unit) Function

The ReLU function, as illustrated in Figure 2.7, returns the input if it's positive, otherwise, it returns zero. It's widely used in hidden layers of neural networks and has been found to accelerate convergence during training. Mathematically,

$$h_{ReLU}(z) = \max(0, z) \quad (2.6)$$

where  $z$  is the pre-activation value.

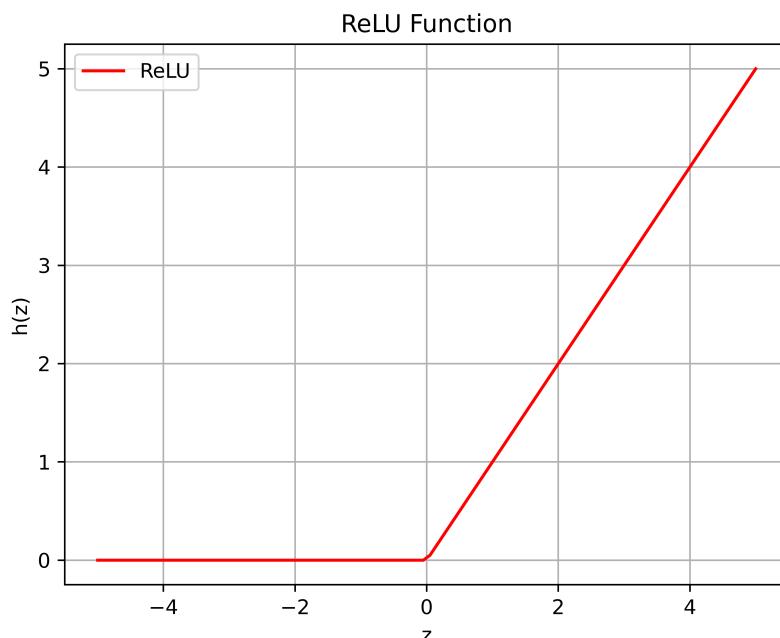


Figure 2.7. Illustration of ReLU Function. Image is generated using Python

ReLU function ကို neural network ၏ hidden layer များအတွက် အဓိက အသုံးပြုသည်။ ဤ function (ညီမှာခြင်း- 2.6)သည် သူည်ထက်ကြီးသည့် input များအတွက် မူလတန်ဖိုးကို ပြန်ထုတ်ပေးသည်။

## 2.2.5 Hyperbolic Tangent (tanh) Function

The hyperbolic tangent function, commonly denoted as  $h_{\text{tanh}}(z)$ , is a mathematical function that maps input values to the range,  $[-1, 1]$  as shown in Figure 2.8. It is similar to the sigmoid function but has a wider range and outputs negative values for negative inputs.

$$h_{\text{tanh}}(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \quad (2.7)$$

where  $z$  is the pre-activation value.

The tanh function is often used in hidden layers of neural networks as an activation function. It has been found to perform well in practice and is particularly useful when the data has zero mean, as it maps negative inputs to negative outputs and positive inputs to positive outputs, preserving the sign of the input.

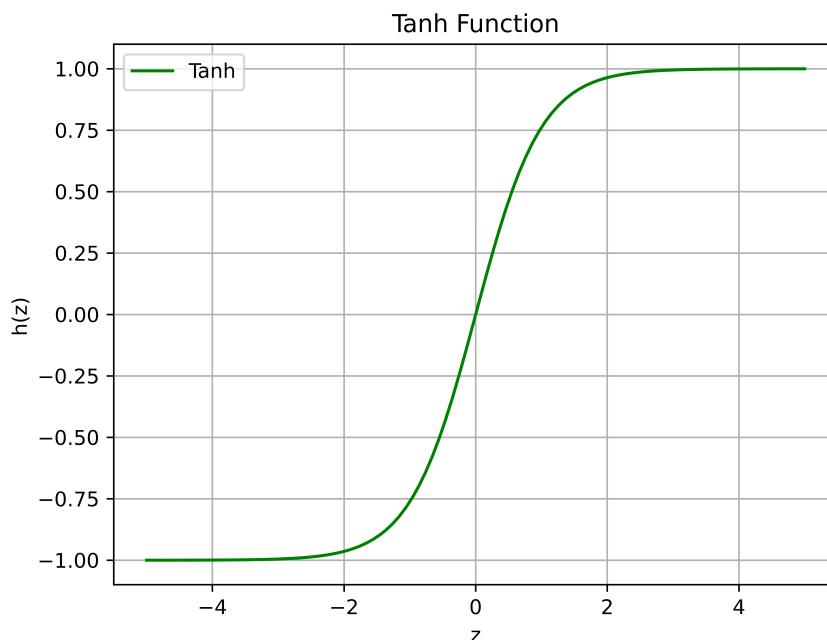


Figure 2.8. Illustration of Hyperbolic Tangent (tanh) Function. Image is generated using Python.

Hyperbolic Tangent (tanh) Function သည် sigmoid Function နှင့် ဆင်တူသည်။ သို့သော် ပုံ (2.8) တွင်ပြသထားသည့် အတိုင်း  $h_{\text{tanh}}(z)$  ၏ တန်ဖိုး သည် အနှစ် တစ်နှင့် အပေါင်းတစ် အကြားတွင် ရှိသည်။ ထိုအပြင် အနှစ်တန်ဖိုးရှိသည့်  $z$  တန်ဖိုးများ အတွက် အနှစ် တန်ဖိုးကို ထုတ်ပေးပြီး သုညထက်ကြီးသည့်  $z$  တန်ဖိုးများအတွက် အပေါင်းတန်ဖိုးကို ထုတ်ပေးရာ မူလ လက္ခဏာ တန်ဖိုးကို ထိန်းသိမ်းထားသည်ဟု ဆိုရမည်။ tanh function သည် လက်တွေ ပုံစံများတွက်ချက်ရာတွင် ရလဒ်ကောင်းများ ရရှိသည် ကို တွေ့ရပြီး hidden layer များအတွက် အမိုက် အသုံးပြုသည်။

## 2.3 Forward Propagation

Forward propagation is the process by which input data passes through the network layers, moving from left to right, to produce an output. It involves a series of mathematical operations that transform the input data, step by step, through each layer of the network. At the start of training, the neural network's weights and biases are typically unknown and initialized randomly.

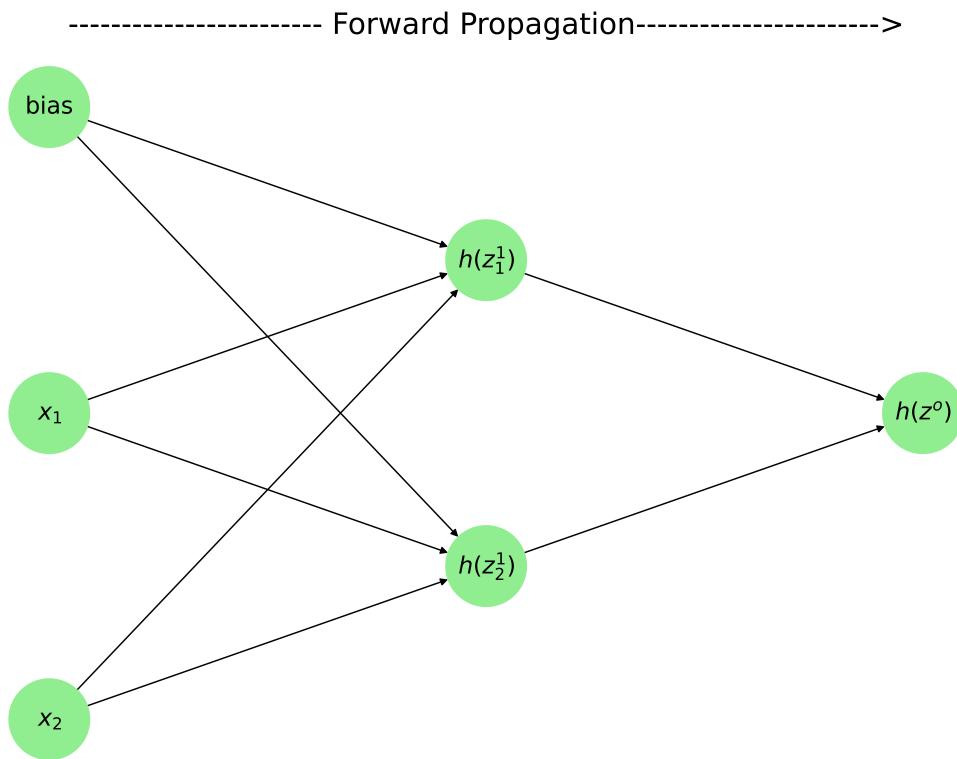


Figure 2.9. Multi-Layer Perception with one hidden layer and two input features. Image is generated using Python networkX Library

Let's consider a simple example with a single hidden layer as shown in Figure 2.9:

- ✿ **Input Layer:** Suppose we have two input features,  $x_1$  and  $x_2$ .
- ✿ **Hidden Layer:** The hidden layer has two neurons. Each neuron receives  $x_1$  and  $x_2$  as inputs and computes the pre-activation values for each neuron,  $z_1^1$  and  $z_2^1$ .

$$z_1^1 = \omega_{11}x_1 + \omega_{12}x_2 + b_1 \quad (2.8)$$

$$z_2^1 = \omega_{21}x_1 + \omega_{22}x_2 + b_2 \quad (2.9)$$

where  $\omega_{ij}$  are the weights associated with the input features and the neurons in the hidden layer, and  $b_i$  are the bias values. The outputs,  $z_1^1$  and  $z_2^1$  are then passed through an activation function and produces:

$$a_1 = h(z_1^1) \quad (2.10)$$

$$a_2 = h(z_2^1) \quad (2.11)$$

- ✿ **Output Layer:** The output layer has one neuron, which takes  $a_1$  and  $a_2$  and computes the output:

$$y = h(z^o) \text{ where } z^o = \omega_{o1}a_1 + \omega_{o2}a_2 + b_o \quad (2.12)$$

$z^o$  is the pre-activation value at the output layer,  $\omega_{oj}$  are the weights associated with the output and the neurons in the hidden layer, and  $b_o$  is the bias value.

In forward propagation, the computed  $y$  represents the predicted value using random weights.

The objective of the machine learning model is to find the parameters (weights and bias) that produce the lowest errors between the actual and predicted values. These parameters are iteratively refined through backward propagation, where the model adjusts its parameters based on the calculated errors to improve its predictive accuracy.

ပုံ 2.3 သည် hidden layer တစ်ခုပါဝင်သော Multi-layer neural network ၏ ဥပမာဖြစ်သည်။ Input layer တွင် feature နှစ်ခု  $x_1$  နှင့်  $x_2$  တို့၏ပြီး အဆိုပါ feature များသည် ကြားခံ layer ၏ Input များဖြစ်သည်။ ထို့နောက်  $z_1^1$  နှင့်  $z_2^1$  တန်ဖိုးကို ညီမျှခြင်း (2.8)ကို အသုံးပြု၍ တွက်ချက်သည်။ တန်လုံးဆိုသော  $z_1^1$  နှင့်  $z_2^1$  ၏ တန်ဖိုးများသည် မြောက်ဖော်ကိန်းများနှင့် မြောက်ထားသည့် input feature များ၏ စုစုပေါင်းတန်ဖိုးဖြစ်သည်။

ထို့နောက် ကြားခံ layer ၏ output များဖြစ်သည့်  $a_1$  နှင့်  $a_2$  ၏ တန်ဖိုးကို activation function တစ်ခုကို သုံး၍ တွက်ချက်ရမည်။ အသုံးများသည့် function များမှာ ReLU, Sigmoid, (သို့မဟုတ်) Tanh တို့ဖြစ်ကြသည်။ အဆိုပါ  $a_1$  နှင့်  $a_2$  သည် Output Layer အတွက် input များဖြစ်သည်။ ထို့နောက် final ရလဒ် ဖြစ်သော  $y$  ၏ တန်ဖိုးကို ညီမျှခြင်း 2.12 ကို အသုံးပြု၍ တွက်ချက်သည်။ Classification ပုံစံများအတွက် Output Layer တွင် softmax activation function ကို အသုံးပြုလေ့ ရှိပြီး regression ပုံစံများ အတွက် linear function ,  $h(z^o) = z^o$  ကို အသုံးပြုကြသည်။

အထက်ပါ ညီမျှခြင်းများ (2.8) 2.12 တွင် မြောက်ဖော်ကိန်းများ ဖြစ်သော  $w_i$  နှင့်  $b_i$  ၏ တန်ဖိုးများကို randomly သတ်မှတ်၍ တွက်ချက်ခြင်း ဖြစ်သည်။ Forward propagation မှ ရရှိသည့်  $y$  ၏ တန်ဖိုးမှာ Multi-layer neural network Model မှ တွက်ချက်ပေးသည့် ခန်းမှန်းတန်ဖိုးသာ ဖြစ်သည်။ Machine learning model များ၏ ရည်ရွယ်ချက်မှာ ခန်းမှန်းတန်ဖိုး နှင့် training ဒေတာ များ၏ မူလတန်ဖိုးကြား ခြားနားချက်ကို အနည်းဆုံးဖြစ်စေသည့် မြောက်ဖော်ကိန်း များ  $w_i$  နှင့်  $b_i$  တန်ဖိုးများကို ရှာရန် ဖြစ်သည်။

## 2.4 Backward Propagation

Backward propagation, also known as backpropagation, involves the iterative adjustment of the model's parameters (weights and biases) based on the calculated errors between the predicted and actual outputs. The Gradient descent is one of the fundamental algorithms used for updating the parameters.

- ✿ **Loss Computation:** After forward propagation, the model's prediction,  $\hat{y}$  is compared to the actual target value  $y$  using a loss function (e.g., mean squared error for regression, binary cross-entropy for binary classification, categorical cross-entropy for multi-class classification). This loss function computes the disparity between the predicted and actual outputs.
- ✿ **Backpropagate Errors:** Once the loss is computed, the algorithm works backward through the network to calculate the gradient of the loss function with respect to each parameter.
- ✿ **Output Layer:** Compute the gradients of weights and bias in the output layer using the chain rule

$$\frac{\delta \text{Loss}}{\delta \omega_o} = \frac{\delta \text{Loss}}{\delta z^o} \cdot \frac{\delta z^o}{\delta \omega_o} \quad (2.13)$$

$$\frac{\delta \text{Loss}}{\delta b_o} = \frac{\delta \text{Loss}}{\delta z^o} \cdot \frac{\delta z^o}{\delta b_o} \quad (2.14)$$

- ✿ **Hidden Layer:** Compute the gradients of weights and bias in the hidden layer

$$\frac{\delta \text{Loss}}{\delta \omega_{\text{hidden}}} = \frac{\delta \text{Loss}}{\delta z^{\text{hidden}}} \cdot \frac{\delta z^{\text{hidden}}}{\delta \omega_{\text{hidden}}} \quad (2.15)$$

$$\frac{\delta \text{Loss}}{\delta b_{\text{hidden}}} = \frac{\delta \text{Loss}}{\delta z^{\text{hidden}}} \cdot \frac{\delta z^{\text{hidden}}}{\delta b_{\text{hidden}}} \quad (2.16)$$

- ✿ **Update Weights and Biases:** Adjust the weights and biases in both the output and hidden layers using the computed gradients and a learning rate. This step involves moving in the opposite direction of the gradients to minimize the loss function.

$$\omega_o^{\text{new}} = \omega_o^{\text{old}} - \text{learning rate} \times \frac{\delta \text{Loss}}{\delta \omega_o} \quad (2.17)$$

$$b_o^{\text{new}} = b_o^{\text{old}} - \text{learning rate} \times \frac{\delta \text{Loss}}{\delta b_o} \quad (2.18)$$

$$\omega_{\text{hidden}}^{\text{new}} = \omega_{\text{hidden}}^{\text{old}} - \text{learning rate} \times \frac{\delta \text{Loss}}{\delta \omega_{\text{hidden}}} \quad (2.19)$$

$$b_{\text{hidden}}^{\text{new}} = b_{\text{hidden}}^{\text{old}} - \text{learning rate} \times \frac{\delta \text{Loss}}{\delta b_{\text{hidden}}} \quad (2.20)$$

Both forward and backward propagation are repeated for multiple **epochs** until the model converges or until a stopping criterion is met.

### Definition 2.1

An **epoch** refers to one complete pass through the entire training dataset. During each epoch, the neural network performs forward and backward propagation (calculating predictions, computing loss, and updating weights) for every sample in the training dataset.

Neural Network Model တစ်ခုကို Training ပြုလုပ်ရာတွင် forward နှင့် backward propagation ကို ကြိမ်ဖန်များစွာ ပြုလုပ်ပြီး ခန့်မှန်းတန်ဖိုးနှင့် မူလတန်ဖိုး၏ ခြားနားချက် အနည်းဆုံးဖြစ်စေမည့်  $\omega_i$  နှင့်  $b_i$  ၏ တန်ဖိုးများကို ရှာဖွေရသည်။ Forward propagation အဆင့်တွင်  $\hat{y}$  ၏ တန်ဖိုးကို တွက်ချက်နိုင်ရန်အတွက် neural network ၏ input layer မှ output layer တဆင့်ချင်း (ဘယ်မှညာသို့) တွက်ချက်ရသည်။

Backward propagation အဆင့်တွင်မူ output layer မှ input layer သို့ (ညာမှ ဘယ်သို့) ပြန်သွားပြီး layer တစ်ခုချင်းစီတွင် ရှိသည့် Parameter များ၏ gradients တန်ဖိုးများကို ညီမျှခြင်း 2.13 နှင့် 2.15 တို့ကို အသုံးပြု၍ တွက်ချက်သည်။ ထို့နောက်  $\omega_i$  နှင့်  $b_i$  များ၏ တန်ဖိုးကို ညီမျှခြင်း 2.17 ကို အသုံးပြု၍ update ပြုလုပ်ပေးရသည်။

Forward နှင့် backward propagation process - ၂ခုလုံးပါဝင်သော Cycle တစ်ခုကို epoch ဟုခေါ်ဆိုပြီး Neural Network Model များ တည်ဆောက်ရာတွင် Programmer/developer မှ epoch အရေအတွက်ကို သတ်မှတ်ပေးရန် လိုအပ်သည်။

## 2.5 Practical Implementation

In this section, three practical projects are presented to reinforce the concepts learned in the previous sections. These projects are designed to provide hands-on experience and deepen your understanding through real-world applications. By working through these projects, you will be able to apply the theoretical knowledge in practical scenarios, thereby solidifying your learning and enhancing your skills.

Walk-through videos are provided via the author's YouTube channel [13]. You can watch the videos while reading and implementing the codes along.

ဤ အခန်း တွင် လက်တွေ့ industry တွင် အသုံးဝင် သည့် project သုံး ခဲကို Neural Network အသုံးပြု၍ တည်ဆောက် ပုံ တည်ဆောက် နည်း အဆင့် ဆင့် ကို ဖော်ပြသွား မည် ဖြစ်သည်။ Deep Learning ကို လေ့လာရာတွင် project များ ကို လက်တွေ့ လိုက်ပါ လေ့ကျင့်ရန် လိုအပ်ပါသည်။ ဤ လေ့ကျင့်ခန်းများ ကို လေ့လာရာတွင် အထောက် အကြဖြစ်စေရန် အတွက် project ကို ရှင်းပြသည့် video များ ကို YouTube Channel တွင် တင်ပေးထားပြီး ဖြစ်သည်။ သက်ဆိုင်ရာ Project အလိုက် ပေးထားသည့် လင့်တွင် သွားရောက် ကြည့်ရှုနိုင်သည်။

### 2.5.1 Regression: Resale House Price Prediction

House price prediction is a significant application of machine learning with substantial implications for various stakeholders in the real estate market. The deployment of these models into commercial platforms enhances decision-making processes, market analysis, and strategic planning. Leveraging large datasets and advanced algorithms, machine learning (ML) models have significantly transformed the landscape of house price prediction, yielding remarkably accurate predictions.

In today's digital landscape, numerous real estate websites and applications integrate ML models to provide price estimates for listed properties. This integration not only elevates user experience and engagement but also empowers buyers and sellers to make informed decisions. Additionally, it enables real estate agents to offer enhanced services and assists investors in identifying opportunities within the market. Continuously improving ML models for higher accuracy is a collaborative effort between academia and industry. Real estate data is dynamic, with various factors influencing prices differently across regions. Therefore, it is essential to develop models tailored to each market's unique characteristics.

In Singapore, an HDB (Housing and Development Board) estate typically refers to a large residential area comprising multiple HDB blocks. Each block is a high-rise building containing numerous housing units, commonly known as HDB flats or apartments. All HDB properties in Singapore typically have a 99-year lease term.

First-time buyers acquire their properties directly from the Singapore government, while resale transactions involve the buying and selling of units among Singapore residents. Each transaction is recorded by the Singapore Housing and Development Board to track ownership. In earlier records, the approval date by HDB is considered the transaction date, but after 2014, the registration date is recorded. These two dates are typically within a month of each other, resulting in minimal fluctuation. However, the resale prices recorded by HDB should be considered indicative only, as the actual resale prices agreed upon between buyers and sellers depend on many factors. Being able to estimate the resale price based on the properties of a flat, such as location, area, and other factors, helps both buyers and sellers make informed decisions.

This section explains, step by step, how to develop a Neural Network (deep learning) model for predicting resale prices of public housing flats (HDB flats) in Singapore.

#### Remark

This work is inspired by a project assignment given to my students during the Supervised Machine Class (iSTAR) of 2024. In this course, students were tasked with developing a regression model to predict housing prices for houses or flats in the region. While the codes and explanations are my original work, the project context was developed as part of the coursework to provide practical learning experiences.

Real estate Online Platform များတွင် Machine learning model များကို ထည့်သွင်းအသုံးပြုလာနိုင်ခြင်းသည် အိမ်ခြံမြေဖော်ကွက်ကို စိတ်ဝင်စားသူများအတွက် အရေးပါသော အရွှေ တစ်ခု ဖြစ်သည်။ ဒေတာ အချက်အလက်များ တနောက်ခြင်း နှင့် အတူ အချက်အလက်မှန်ကန်သည့် ဒေတာများကို အစိုးရ website များ အပါအဝင် အဖွဲ့အစည်းအပျိုးမျိုးက အများပြည် သူ အသုံးပြုနိုင်အောင်ဖွင့်ပေးလာကြသည်။ ထိုသို့ ဒေတာများ အလွယ်တကူ ရရှိနိုင်ခြင်း ကွန်ပြုတာများ အဆင့်မြှင့်လာခြင်းတို့သည် အိမ်ခြံမြေ နှင့် တိုက်ခန်းများ၏ ဈေးနှုန်းခန်းများရန်အတွက် ပိုမိုကောင်းမွန်သည့် Machine learning model များ တည်ဆောက်ရန် အထောက်အပံ့ကောင်းများ ဖြစ်သည်။

အိမ်ခြံမြေဈေးကွက်၏ သဘော သဘာဝမှာ နေရာအသေတစ်ခုနှင့် တစ်ခု အပေါ်မှတ်သူ၏ အချက်အလက်များ ကွဲပြားကြသည်။ သို့ဖြစ်ရာ Machine learning model တစ်ခုထဲကို နေရာအမျိုးမျိုးတွင် အသုံးပြုရန်မှာ မဖြစ်နိုင်ပေ။ နိုင်ငံ (သို့ မဟုတ်) မြို့နယ် အလိုက် ရရှိနိုင်သော ဒေတာအချက်အလက်များကို အသုံးပြု၍ တည်ဆောက်မှသာလျှင် ပို၍ တိကျသော ဈေးနှုန်းခန်းများချက်များကို ရရှိနိုင်မည် ဖြစ်သည်။

Singapore နိုင်ငံတွင် အများပြည်သူများ သက်သက်သာသာ နှင့် ဝယ်ယူ နေထိုင်နိုင်ရန် အစိုးရအဖွဲ့အစည်းတစ်ခုဖြစ်သည့် HDB (Housing and Development Board) မှ တာဝန်ယူ ဆောက်လုပ်သည့် တိုက်ခန်းများကို HDB တိုက်ခန်းများဟု ခေါ်ဆိုကြသည်။ မြို့နယ် အလိုက် HDB အိမ်ယာများစွာ ရှိပြီး အိမ်ယာတစ်ခုတွင် အထပ်မြှင့် အဆောက်အအုံတစ်ခု မက ပါဝင်သည်။ အဆိုပါ အဆောက်အအုံများကို Singapore အစိုးရမှ ပိုင်ဆိုင်ပြီး ပြည်သူကို ရောင်းချသည့် သက်တမ်းမှာ ဤနှစ်သာ ဖြစ်သည်။ တိုက်သက်တမ်းလွန်သည့် တိုက်ခန်းများကို အစိုးရက ဖြေဖျက်ပြီး အသစ်ပြန်လည်ဆောက်လုပ်လေ့ရှိသည်။ အသစ်ဆောက်လုပ်ပြီးစီသည့် HDB တိုက်ခန်းများကို Singapore အစိုးရထံမှသာ တိုက်ရှိပါ ဝယ်ဆိုနိုင်သည်။ သတ်မှတ် ကာလတစ်ခု (များသောအားဖြင့် ၅ နှစ်) ကျော်လွန်သွားပါက HDB တိုက်ခန်းဟောင်းများကို နေထိုင်သူ အချင်းချင်းကြား ပြန်လည်ရောင်းချခြင်း၊ ဝယ်ယူခြင်းများ ပြုလုပ်နိုင်သည်။

သို့သော် အရောင်းအဝယ် ပြုလုပ်မည်ဆိုပါက Singapore အစိုးရ (HDB) ထံမှ တရားဝင် ခွင့်ပြုချက်ရယူရန်လိုအပ်သည်။ အဆိုပါ အရောင်းအဝယ်မှတ်တမ်းတိုင်းကို HDB တွင် စနစ်တိကျ စာရင်းသွင်းထားရှိသည်။ အစောပိုင်း စာရင်းသွင်းချက်များတွင် HDB မှ ခွင့်ပြုပေးလိုက်သည့် နေဂုံးရှိ အရောင်းအဝယ်ပြုလုပ်သည့်နေဂုံးဟု သတ်မှတ်ခဲ့ကြသော်လည်း နောက်ပိုင်း မှတ်တမ်းများတွင်မူ HDB တွင် မှတ်တမ်းတင်သည့်နေဂုံးရှိ အရောင်းအဝယ်ပြုလုပ်သည့်နေဂုံးဟု သတ်မှတ်သည်။ ယေဘုယျအားဖြင့် HDB မှ ခွင့်ပြုပေးလိုက်သည့် နေဂုံးမှတ်တမ်းတင်သည့်နေဂုံးမှာ တစ်လအတွင်းဖြစ်သဖြင့် ကြားထဲတွင် ဈေးနှုန်း အပြောင်းအလဲ ဖြစ်လေ့မရှိပါ။ သို့သော် ရောင်းသူနှင့် ဝယ်ယူ အကြားသဘောတူညီချက်များကို မှတ်သူ၏ HDB ၏ မှတ်တမ်းရှိ ဈေးနှုန်းများ ပြင်ပပေါက်ဈေးများသည် အနည်းငယ် ကွာဟနိုင်သည်။

ဤသင်ခန်းစာတွင် Singapore နိုင်ငံရှိ HDB တိုက်ခန်းများ၏ ဈေးနှုန်းကို ခန်းများသည် Neural Network Model တစ်ခုအား Keras Library ကို အသုံးပြု၍ တည်ဆောက်သည့် အဆင့်ဆင့်ကို ရှင်းပြုသွားမည်။

## Dataset

The dataset used for this project is obtained from the Singapore government's open data portal [14], which provides various datasets relevant to different aspects of public administration, urban planning, health, transportation, and other sectors managed by the government. Using a reliable data source is crucial for building an accurate and robust machine learning model.

The dataset includes detailed information on resale transactions of HDB flats between January 1, 2017, and March 30, 2024. The dataset contains 175,672 rows and 11 columns and was downloaded on May 23, 2024. Further details regarding the dataset are outlined in Table 2.1.

ယခု Project တွင် အသုံးပြုသည့် ဒေတာကို Singapore အစိုးရ၏ open data portal [14] မှ ရယူထားခြင်းဖြစ်သည်။ အဆိုပါ website ကို Singapore အစိုးရ အဖွဲ့အစည်းတစ်ခုဖြစ်သည့် GovTech မှ တည်ထောင်ထားခြင်းဖြစ်ပြီး Singapore ဝန်ကြီးဌာန အချို့နှင့် အစိုးရ အဖွဲ့အစည်းများ၏ ဒေတာများကို ပြည်သူအများ အသုံးပြုနိုင်ရန် တင်ပေးထားသည်။ AI နှင့် Machine Learning Project များ ပြုလုပ်ရာတွင် ယုံကြည် စိတ်ချရသည့် ဒေတာ အချက်အလက်များကို ရယူနိုင်ရန် အရေးကြီးသည်။

ဤ dataset ကို ၂၀၂၄ ခုနှစ် မတ်လ ၂၃ ရက်နေ့တွင် အထက်ပါ website မှ download ရယူခဲ့ပြီး ငါးdataset တွင် ၂၀၁၇ နေ့ရိုက်လ ၁ ရက်မှ ၂၀၂၄ မတ်လ ၃၁ ရက်နေ့အထိ ရောင်းချထားသော Singapore HDB တိုက်ခန်းများ၏ အရောင်းစာရင်းများ ပါဝင်သည်။ စုစုပေါင်းပါဝင်သည့် အရောင်းတိုက်ခန်း အရေအတွက် မှာ တစ်သိန်း ခုနှစ်သောင်း ငါးထောင် ခြောက်ရာ ခုနှစ်ဆယ့် နှစ်ခု (၁၇၅,၆၇၂) ဖြစ်ပြီး ရောင်းချမှုအတွက် စာရင်းသွင်းသည့် ခုနှစ်နှင့်လာ တိုက်ခန်း အကျယ်အဝန်း၊ အမျိုးအစား၊ တည်နေရာ စသည်ဖြင့် Column -၁၁ ခု ပါဝင်သည်။ အထက်ပါ ယေဘုယျအားဖြင့် ဖော်ပြထားသည့် dataset တွင် ပါဝင်သော အချက်အလက်များ၏ အဓိပ္ပာယ် ဖွင့်

Table 2.1. Data Attributes of Singapore HDB Flat (January - 2017 to March - 2024)

Num	Data Attributes	Column Name	Data Type	Description
1	Month	month	Text (Date-Time)	Month and Year of sale
2	Town	town	Text	Designated residential area
3	Flat type	flat_type	Text	Classification of units by room size
4	Block	block	Text	The Block number where the unit sold located
5	Street name	street_name	Text	Street name of the unit sold located
6	Storey range	storey_range	Text	Estimated range of floors the unit sold was located on
7	Floor area sqm	floor_area_sqm	Numeric	Total interior space within the unit, measured in square meters
8	Flat model	flat_model	Text	Classification of units by generation
9	Lease commence date	lease_commence_date	Numeric	Starting point of a lease agreement (Year)
10	Remaining lease	remaining_lease	Text	Remaining amount of time left on the lease (Years and Months)
11	Resale price	resale_price	Numeric	Resale Price of the flat sold

ဆုံးချက်အသေးစိပ်မှာ အောက်ပါအတိုင်း ဖြစ်သည်။

- ❖ month-> တိုက်ခန်း ရောင်းချေမှုကို စာရင်းသွင်းသည့် ခုနှစ်၊ လ။
- ❖ town-> တိုက်ခန်း တည်ရှိသည့် မြို့အမည်။
- ❖ Flat type -> တိုက်ခန်း အမျိုးအစား - ပါဝင်သည့် အခန်းအရေအတွက်ကို မူတည်၍ သတ်မှတ်ပြီး အမျိုးအစား ဂ ခ ရှိသည်။
- ❖ Block-> တိုက်ခန်း တည်ရှိသည့် အဆောက်အအုံ၏ နံပါတ်။
- ❖ Street name-> တိုက်ခန်း တည်ရှိသည့် လမ်းအမည်။
- ❖ Storey range-> တိုက်ခန်း တည်ရှိသည့် အလွှာ/အထပ် အုပ်စု။
- ❖ Floor area sqm-> တိုက်ခန်း၏ ဧရိယာ အကျယ်အဝန်း ( square meter နှင့် ပေးထားသည်။)
- ❖ Flat model-> တိုက်ခန်း၏ မော်ဒယ် (တည်ဆောက်သည့် ပုံစံကို မူတည်၍ သတ်မှတ်ပြီး အမျိုးအစား ဂါ မျိုး ရှိသည်။)
- ❖ Lease commence date-> Housing and Development Board မှ စတင်ရောင်းချေသည့် ရက်စွဲ။
- ❖ Remaining lease-> တိုက်ခန်း၏ လက်ကျန် သက်တမ်း (အစိုးရ ခွင့်ပြုသည့် သက်တမ်းမှာ ဤ နှစ်ဖြစ်သည်။)
- ❖ Resale price-> စာရင်းသွင်းသည့် ရောင်းချေ ဈေးနှုန်း။

### ☞ Remark

တိုက်ခန်း ရောင်းချေမှုကို HDB တွင် စာရင်းသွင်းသည့် ရက် နှင့် အမှန် ရောင်းချေမှုသည် များ သော အား ဖြင့် တစ် လ အတွင်း ဖြစ်သည်။

## Data Preprocessing

The dataset is well maintained and prepared by the Singapore government's open data portal [14]. There is no missing data, which simplifies the preprocessing steps.

ဤ dataset သည် Singapore အစိုးရ၏ မှ data portal ရယူထားသည့်ဖြစ်ရာ ဒေတာများမှာ အမှားအယွင်းကင်းပြီး အချက်အလက် ပြည့်စုံသည်။ သို့သော အချို့ အချက်အလက်များကိုမူ Neural Network (deep learning model) တစ်ခု တည်ဆောက်ရန်အတွက် အဆင့်မြင့် ဖြစ်စေရန် ပြင်ဆင်မှု အချို့ ပြုလုပ်ရသည်။

## Data Preprocessing: Data Cleaning

- In the given dataset, **Lease commencement date** and **remaining lease** are two attributes conveying almost identical information. The former signifies the year when the lease for the property commenced. The latter denotes the number of years remaining on the lease at the time of the transaction. Since all HDB properties in

Singapore typically have a 99-year lease term, these two attributes are related by:

$$\text{remaining lease} = 99 - (\text{sold year} - \text{lease commencement year}) \quad (2.21)$$

As a result, the **Lease commencement date** attribute is dropped to avoid redundancy.

2. Similarly, three features relate to the location of the flat: '**block number**', '**street name**', and '**town**'. While '**block number**', and '**street name**' provide the exact location of the resale flat, these three features are highly correlated. Hence, only the '**town**'. attribute is considered, and the other two attributes are dropped.

ဤ dataset တွင် တိုက်ခန်း၏ သက်တမ်းနှင့် ပတ်သက်၍ အချက် ၂ ခု ပေးထားသည့်ကို တွေ့ရသည်။ ပထမ အချက် ဖြစ်သော **Lease commencement date** သည် တိုက်ခန်းကို အစိုးရမှ စတင်ရောင်းချသည့် ရက်စွဲဖြစ်ပြီး အခြား အချက်မှာ မူတိုက်ခန်း၏ လက်ကျွန်း သတ်တမ်း **remaining lease** ဖြစ်သည်။ အစိုးရ သတ်မှတ်ထားသည့် တိုက်ခန်း၏ သက်တမ်းမှာ ၉၉ ခုနှစ် ဖြစ်ရာ အဆိုပါ တိုက်ခန်းကို လက်လွှာရောင်းချချိန်တွင် ရှိမည့် တိုက်ခန်း၏ လက်ကျွန်း သက်တမ်းသည် ၉၉ နှစ် ထဲမှ လက်ရှိသက်တမ်းကို နှိုတ်ခြင်း ဖြစ်သည်။ ဥပမာ – ၁၉၉၄ တွင် အစိုးရမှ စတင်ရောင်းချ သော တိုက်ခန်းများ၏ သက်တမ်းသည် ၂၀၂၄ ခုနှစ် တွင် သက်တမ်းနှစ် (၃၀) ရှိနေပြီ ဖြစ်သည်။ သို့ဖြစ်ရာ အဆိုပါ တိုက်ခန်းကို ယခု အချိန် (၂၀၂၄ ခုနှစ်) တွင် ရောင်းချမည့်ဆိုပါက တိုက်ခန်း၏ လက်ကျွန်း သက်တမ်းမှာ ၆၉ နှစ် (၉၉-၃၀) ဖြစ်သည်။

ဤ Project အတွက် Neural Network (deep learning model)ကို တည်ဆောက်ရာတွင် တိုက်ခန်း၏ လက်ကျွန်း သက်တမ်း **remaining lease** ကိုသာ ထည့်သွင်းစဉ်းစားမည်။

ထိုအတူ တိုက်ခန်း၏ တည်နေရာ ကို အဆောက်အအုံ၏ နံပါတ်၊ လမ်းအမည်၊ မြို့အမည် – စသည် ဖြင့် အချက်အလက် ၃ ခု ဖြင့် ပေးထားသည်။ အဆိုပါ အချက်အလက် များမှာ လိပ်စာကို အတိအကျ သိရန် အရေးကြီးသော်လည်း တိုက်ခန်း၏ ဈေးနှုန်းကို သတ်မှတ်ရန် အတွက်မူ သဘောတရားတူညီနေသည့် အချက်များ ဖြစ်သည်။ သို့ဖြစ်ရာ ဤ Project တွင် တိုက်ခန်း၏ မြို့အမည်ကို သာ ထည့်သွင်းစဉ်းစားမည်။

## Data Preprocessing: Categorical Data Encoding

1. The *flat type* attribute consists of 7 categories representing different sizes of flats. As this data is ordinal, it is encoded using label encoding to maintain the order of the categories based on the size of the flat.
2. The *flat model* attribute includes 21 categories, which do not have a natural order. Therefore, one-hot encoding is applied to encode this data.
3. The *town* attribute identifies the estate in which the flat is located, with 26 unique locations. As there is no inherent order or relationship among these locations, one-hot encoding is also applied to this attribute.
4. The *storey range* attribute describes the range of storeys that a flat resale falls into, such as "10 TO 12" or "01 TO 03". To simplify this categorical attribute, the range is replaced by the median value (e.g., "10 TO 12" is replaced by "11").

ဤ dataset တွင် ပါဝင်သည့် အချို့ အချက်အလက် များသည် စာသားများဖြင့် ဖော်ပြထားသည့် အပ်စုပြ ဒေတာများ ဖြစ်သည်။ ဥပမာ – တိုက်ခန်းအမျိုးအစား၊ မော်ဒယ်၊ မြို့နယ် တိုက်ခန်း၏ အလွှာ အပ်စု တို့ ဖြစ်သည်။ Neural Network (deep learning model) တစ်ခု တည်ဆောက်ရာတွင် အသုံးပြုမည့် ဒေတာအချက်အလက် များမှာ ကိန်းကဏ္ဍားများ ဖြစ်ရန် လိုအပ်သည်။ သို့ဖြစ်ရာ အဆိုပါ အပ်စုပြ ဒေတာအမျိုးအစားများကို ကိန်းကဏ္ဍားများဖြင့် ဖော်ပြမည်။

တိုက်ခန်းအမျိုးအစား ( *flat type* ) သည် တိုက်ခန်းအကျယ်အဝန်းပေါ်မူတည်၍ သတ်မှတ်ခြင်းဖြစ်ပြီး အမျိုးအစား ( ဂ ) မျိုးရှိသည် ကို တွေ့ရသည်။ အခန်း ( ဂ ) ခန်းပါဝင်သော တိုက်ခန်းသည် အခန်း ( ခ ) ခန်းပါဝင်သော တိုက်ခန်းထက် ပို၍ ကျယ်ဝန်းသည်။ သို့ဖြစ်ရာ တိုက်ခန်းအမျိုးအစား ( *flat type* ) ကို ကိန်းကဏ္ဍားပြောင်းရာတွင် label encoding ဟု ခေါ်သည့် တိုက်ရှိက် ပြောင်းသည့် စနစ်ကို အသုံးပြုမည်။ အငယ်ဆုံးတိုက်ခန်းအမျိုးအစားကို - တစ်ဟု သတ်မှတ်ပြီး အကျယ်ဆုံး အမျိုးအစားကို ခုနှစ် ( ဂ ) ဟုတ် သတ်မှတ်မည်။

တိုက်ခန်း၏ မော်ဒယ် ( *flat model* ) သည် တည်ဆောက်သည့် ခုနှစ်၊ ဒီဇိုင်းပေါ်မူတည်၍ သတ်မှတ်ခြင်း ဖြစ်ပြီး အမျိုးအစား ပေါင်း ၁၁ မျိုး ရှိသည်ကို တွေ့ရသည်။ သို့ဖြစ်ရာ တိုက်ခန်း၏ မော်ဒယ် ( *flat model* ) ကို ကိန်းကဏ္ဍားပြောင်းရာတွင် one-hot encoding ဟု ခေါ်သည့် Binary column များအဖြစ် ပြောင်းသည့် စနစ်ကို အသုံးပြုမည်။ ငြင်း one-hot encoding စနစ်တွင် အမျိုးအစား အရေအတွက်အလိုက် Binary column များ သတ်မှတ်ပြီး အမျိုးအစား အပေါ်မူတည်၍ column တစ်ခု၏ တန်ဖိုးမှာ တစ်ဖြစ်ပြီး ကျွန်း column များ၏ တန်ဖိုးမှာ သူညီဖြစ်မည်။

ထိုအတူ တိုက်ခန်း တည်ရှိသည့် မြို့ (town) သည်လည်း အပ်စုပြ ဒေတာဖြစ်ပြီး Singapore နိုင်ငံတွင် မြို့ပေါင်း ၂၆ မြို့ရှိသည်။ အဆိုပါ မြို့များကိုလည်း one-hot encoding ကို အသုံးပြု၍ ကိန်းကဏ္ဍားအဖြစ် ပြောင်းလဲမည်။ ဥပမာ - Jurong East ရှိတွင် တည်ရှိသည့် တိုက်ခန်းအတွက် Jurong East column ၏ တန်ဖိုးမှာ တစ်ဖြစ်ပြီး ကျန် မြို့များကို ပြသည့် column များမှာ သူည့်ဖြစ်မည်။

နောက်ဆုံး အပ်စုပြ ဒေတာ တစ်ခုမှာ တိုက်ခန်း၏ အလွှာ အပ်စုပြသည့် column ဖြစ်သည်။ ဤ dataset တွင် တိုက်ခန်း၏ အလွှာကို ဖော်ပြရာတွင် မူလ နံပါတ် အစား အပ်စုဖြင့်ဖော်ပြထားသည်။ ဥပမာ - ၁၀၊ ၁၁နှင့် ၁၂ လွှာတွင် ရှိသည့် တိုက်ခန်း အားလုံး ကို "10 TO 12" အပ်စုဟု ဖော်ပြသည်။ သို့ဖြစ်ရာ အဆိုပါ အချက်အလက်ကို ကိန်းကဏ္ဍားပြောင်းရန်အတွက် အလယ်အလွှာကို သုံး၍ ဖော်ပြမည်။ ယခု သတ်မှတ်ချက်တွင် ၁၀၊ ၁၁နှင့် ၁၂ လွှာတွင် ရှိသည့် တိုက်ခန်း အားလုံးကို "10 TO 12" အပ်စုဟု ဖော်ပြမည် အစား ကိန်းကဏ္ဍားဖြင့် ၁၁ လွှာသာ ဖော်ပြမည်။

## Data Preprocessing: Feature Engineering

Feature engineering is the process of transforming raw data into meaningful features that enhance the performance of machine learning models. In this project, we perform feature engineering on two features:

1. The **remaining lease** attribute is originally presented in years and months (e.g., 21 years and 6 months), and is subsequently transformed into the equivalent number of months.

$$\text{remaining lease months} = 12 * (\text{remaining lease}[year] + \text{remaining lease}[months]) \quad (2.22)$$

2. Figure 2.10 illustrates a strong correlation between resale prices and transaction months, depicted by the blue-colored line. While this correlation provides valuable insights, relying on the provided resale price for predictions may not be ideal, as it primarily reflects the price inflation. To mitigate the influence of inflation on resale prices, normalization techniques are employed. The HDB resale price index [15], which tracks overall price movements in the public residential market, is utilized for this purpose. Resale prices are adjusted using Equation 2.23 to account for inflation.

$$\text{Adjusted Price} = \frac{\text{RPI}_{2024} \times \text{Resale Price}}{\text{RPI-at-Transaction}} \quad (2.23)$$

where **RPI<sub>2024</sub>** refers to the resale price index for Quarter 1 (Jan-March) 2024, and **RPI-at-Transaction** refers to the resale price index at the time of the transaction. Comparing the blue line and yellow line in Figure 2.10, it can be observed that the new feature '**Adjusted Price**' effectively reduces the inflation effect. This adjustment helps to better understand the resale price of a flat by considering factors such as size, type, and location of the flat. As a result, the **month** and **resale price** attributes are dropped and the inflation '**Adjusted Price**' is used as a target, in favour of predicting the resale price of a flat at the value of the Singapore dollar at Q1 2024.

Feature engineering ဆိုသည့်မှာ မူလ dataset တွင် ပါဝင်သည့် အချက်အလက်များကို မှတည်၍ အချက်အလက် အသစ်များ ဖန်တီးခြင်း၊ မူလ အချက်အလက်များကို machine learning မော်ဒယ် တည်ဆောက်ရာတွင် ပိုမို အဆင်ပြောစေရန် ပြုပြင်ခြင်းများ ကို ခေါ်သည်။ မူလ dataset တွင် တိုက်ခန်း၏ လက်ကျန်သက်တမ်း (**remaining lease**) ကို ဖော်ပြရာ၌ ခုနှစ်နှင့်လ - ၂ ခုလုံးကို ဖော်ပြသည်။ အဆိုပါ အချက်အလက်ကို တွက်ချက်မှု ပြုလုပ်ရာတွင် လွယ်ကူစေရန်အတွက် စုစုပေါင်းလဲဖြင့် ဖော်ပြမည်။ ဥပမာ - သက်တမ်း ၄၁ နှစ် ၆ လ - ကျန်သည့် တိုက်၏ လက်ကျန်သက်တမ်းကို ဖော်ပြရာတွင် ၄၉၈ လ ဟု လ အရည်အတွက်နှင့် ဖော်ပြသွားမည်။

အထက်ပါပုံတွင် တိုက်ခန်း၏ ဈေးနှုန်းများကို လ အလိုက် နှုန်းယဉ်ကြည့်ရာ (အပြာရောင်လိုင်းဖြင့် ဖော်ပြထားသည်) တိုက်ခန်း များ၏ ဈေးနှုန်းသည် အချိန် အလိုက် ပြောင်းလဲ နေသည် ကို တွေ့ရမည်။ အထူးသဖြင့် ၂၀၂၁ နောက်ပိုင်း တွင် သိသုတေသန ဈေးနှုန်းများ ပြန်လည် မြှင့်တက်လာသည် ကို တွေ့နှင့်သည်။ အဆိုပါ အချက်သည် အနာဂတ်တွင် ဖြစ်ပေါ်မည့် ဈေးနှုန်းကို ခန့်မှန်းရာတွင် အသုံးဝင်သည့် အချက်ဖြစ်သော်လည်း တိုက်ခန်း၏ အခြားအချက်အလက်များဖြစ်သည့် အကျယ်အဝန်း တည်နေရာ စသည့် အချက်အလက်များကို မှတည်၍ စွဲစွဲစားရာတွင်မူ သတိထားရမည့် အချက် ဖြစ်လာသည်။ ဥပမာ - ၂၀၂၃ တွင် ၅ သိန်း ၅ သောင်းနှင့် ရောင်းချွဲချွဲသည့် တိုက်ခန်းကို ၂၀၂၄ ခုနှစ်တွင် ရောင်းချွဲဆိုပါက ၆ သိန်း နီးပါးဖြင့် ရောင်းချွဲမည် ဖြစ်သည်။ သို့ဖြစ်ရာ တိုက်ခန်း၏ ဈေးနှုန်းကို တွက်ချက်ရာတွင် မူလ ပေးထားသည့် ဈေးနှုန်းအတိုင်း တိုက်ရှိက်တွက်ချက်ခြင်း မပြုဘဲ ၂၀၂၄ ခုနှစ်တွင် ဖြစ်ပေါ်မည့် ခန်းမှုန်း ဈေးနှုန်းကို မှတည်၍ စွဲစွဲစားသွားမည်။ ၂၀၂၄ ခုနှစ် ပထမ ၃ လ (နေ့နာရီရီမှ မတ်လ) အတွင်း ဖြစ်နိုင်သည့် ဈေးနှုန်းကို သိရှိရန်



Figure 2.10. Comparative illustration: Resale Price vs. Adjusted Price - Mitigating Inflationary Effects

အတွက်မှ Singapore အစိုးရမှ ထုတ်ပြန်ထားသည့် HDB resale price index ကို အသုံးပြချု ညီမျှခြင်း 2.23 တွင် ဖော်ပြထားသည့် အတိုင်း ခန့်မှန်းသွားမည် ဖြစ်သည်။

## Model Development: Build a Neural Network using Keras

After data preprocessing, the clean dataset maintains its original size with 175,672 rows and now consists of 52 columns. The target variable is the ***inflation-adjusted price***, while the remaining 51 columns serve as features for training the neural network (deep learning) model. This section provides an explanation of each step in building the model.

အထက်ပါ data preprocessing အဆင့်ဆင့်ကို ပြလုပ်ပြီးနောက် dataset အသစ်တွင် အရောင်း စာရင်း အရေအတွက်မှာ မူလ အတိုင်း တစ်သိန်း ခုနှစ်သောင်း ငါးထောင် ခြောက်ရာ ခုနှစ်ဆယ့် နှစ်ခု (၁၇၅,၆၇၂) ဖြစ်သော်လည်း column အရေအတွက်မှာမူ ၅၂ ခု အထိတိုးလာသည်။ inflation ကို ထည့်သွင်းစဉ်းစားထားသည့် ဈေးနှုန်းကို Target အဖြစ်ထား၍ ခန့်မှန်းမည် ဖြစ်ပြီး ကျော် column ၅၁ ခု ကို Neural Network (deep learning model) ၏ input များ အဖြစ် အသုံးပြုမည်။

### Step 2.5.1.1

The project starts by importing the pandas library and loading a cleaned CSV file named ***SGHDB2017-2024\_clean.csv*** into a Pandas DataFrame ***df***. Then, the ***adjusted\_price*** column, which represents the housing prices to be predicted, is extracted and stored in the variable ***y*** and the remaining columns, which serve as features for the model, are stored in the variable ***X***.

အဆင့် (၁) တွင် ဦးစွာ ပထမ dataset အသစ် ကို Pandas DataFrame ***df*** အဖြစ် သိမ်းဆည်းခြင်း ဖြစ်သည်။ ထိုနောက် အသုံးပြုမည့် Target ကို variable ***y*** အဖြစ်လည်းကောင်း၊ ကျော် column များကို variable ***X*** အဖြစ်လည်းကောင်း သိမ်းဆည်း မည်။

```
# =====#
import pandas as pd
df = pd.read_csv('SGHDB2017-2024_clean.csv')
y = df['adjusted_price'].values
X = df.drop(columns = 'adjusted_price')
# =====#
```

**☞ Remark**

Ensure you have the necessary libraries installed. You can install them using pip if they are not already installed. For example:

```
pip install tensorflow pandas scikit-learn
```

သတိပြုရမည့် အချက်မှာ လိုအပ်သည့် library များကို install ပြုလုပ်ရန် ဖြစ်သည်။ အကယ်၍ install မပြုလုပ်ရသေးပါက အထက်ပါ ကုဒ်ကို အသုံးပြု၍ install ပြုလုပ်ပေးရမည်။

**Step 2.5.1.2**

This step prepares the data for building the machine learning model.

- ❖ The data is first split into training and testing sets, with 70% of the data used for training ( $X_{train}$  and  $y_{train}$ ) and 30% for testing ( $X_{test}$  and  $y_{test}$ ). The `random_state=42` ensures reproducibility of the split.
- ❖ The continuous features are standardized using `StandardScaler`, which scales the data to have a mean of 0 and a standard deviation of 1. This transformation is applied separately to the training and testing sets to prevent data leakage. In this dataset, there are four continuous features: `flat_type`, `floor_area_sqm`, `floor`, and `remaining_lease_months`.
- ❖ The scaled continuous features are combined with the binary features to form the final training ( $X_{train}$ ) and testing ( $X_{test}$ ) sets.

ဒုတိယ အဆင့်တွင် machine learning model တည်ဆောက်ရန် ဒေတာ အချက်အလက်များကို ပြင်ဆင်ရမည် ဖြစ်သည်။ ဤဗျာ ပထမ datasetကို training အတွက် ဂုဏ်ရေး ရာခိုင်နှုန်း ကို အသုံးပြု၍ ပြီး ကျိုး ၃၀ ရာခိုင်နှုန်း ကို testing အတွက် ချုံထား မည် ဖြစ်သည်။ ထိုနောက် continuous features များ ဖြစ်သည့် `flat_type`, `floor_area_sqm`, `floor`, နှင့် `remaining_lease_months` များကို `StandardScaler` method ကို အသုံးပြု၍ range တစ်ခုအတွင်းရှိအောင် ပြုလုပ်ရမည်။ ထိုသို့ ပြုလုပ်ရတွင် data leakage မဖြစ်စေရန် training နှင့် testing ဒေတာ များကို သီးသန်းပြုလုပ်ရမည် ဖြစ်သည်။

```
# =====#
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
# Step 2: Prepare the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardize continuous features
continuous_columns = ['flat_type', 'floor_area_sqm',
                      'floor', 'remaining_lease_months']
binary_columns = df.columns.difference(continuous_columns
                                       + ['adjusted_price']).tolist()
scaler = StandardScaler()
X_train_continuous = scaler.fit_transform(X_train[continuous_columns])
X_test_continuous = scaler.transform(X_test[continuous_columns])

# Combine scaled continuous features and binary features
X_train = np.hstack([X_train_continuous, X_train[binary_columns].values])
X_test = np.hstack([X_test_continuous, X_test[binary_columns].values])
# =====#
```

**Step 2.5.1.3**

This step defines a function `create_regression_model` that constructs a feedforward neural network model for regression using TensorFlow's Keras API.

တတိယ အဆင့်သည် TensorFlow၏ Keras API ကို အသုံးပြု၍ neural network model တစ်ခု တည်ဆောက် ခြင်းဖြစ်သည်။ ယခု model တွင် hidden layer - ၂ ခုသာ ပါဝင်မည်ဖြစ်ပြီး ပထား hidden layer တွင် neuron - ၃၂ ခု နှင့် ဒုတိယ hidden layer တွင် neuron - ၁၆ ခု ကို အသုံးပြု မည်။ layer - ၂ ခုလုံးအတွက် ReLU activation function ကို အသုံးပြု ထားသည်။

- ❖ This function takes `input_shape` as a parameter, representing the shape of the input data.
- ❖ Inside the function, a sequential model, as illustrated in Figure 2.3, is created using `tf.keras.Sequential()`.
- ❖ The model comprises an input layer, two hidden layers, and an output layer:
  - ❖ The input layer serves as the initial receiver of input data, passing it to subsequent layers for processing.
  - ❖ The first hidden layer is a fully connected (dense) layer with 32 units and ReLU activation function, as depicted in Figure 2.7.
  - ❖ The second hidden layer is also fully connected, with 16 units and ReLU activation function.
  - ❖ These hidden layers learn intricate patterns among the features.
  - ❖ The output layer has a single unit, representing the regression prediction.
- ❖ The code `model.compile(optimizer='adam', loss='mean_squared_error')` configures the neural network model. This project employs the Adam optimizer, an adaptive learning rate optimization algorithm. Adam adjusts the learning rate during training, facilitating faster convergence and improved performance. Additionally, '`mean_squared_error`' is used as the loss function, quantifying the difference between predicted and actual outputs during training.

```
# =====#
import tensorflow as tf
def create_regression_model(input_shape):
    model = tf.keras.Sequential([
        tf.keras.layers.InputLayer(shape=input_shape),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(16, activation='relu'),
        tf.keras.layers.Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')

    return model
# =====#
```

**☞ Remark**

The size of hidden layers and the number of layers in a neural network are critical hyperparameters that influence the model's capacity, performance, and generalization ability. Finding the optimal configuration often involves a balance between model complexity and simplicity, guided by the specific characteristics of the dataset and the requirements of the task at hand.

Hidden layer အရေအတွက်နှင့် neuron အရေအတွက်တို့မှာ neural network တစ်ခုအတွက် အရေးကြီးသည့် hyperparameter များဖြစ်ပြီး dataset နှင့် ဖြေရှင်းမည့် ပြဿနာပေါ်တွင် မူတည်သည်။

**Step 2.5.1.4**

The model is trained using the training data ( $X_{train}$  and  $y_{train}$ ). The training process runs for 10 epochs (Refer definition in Section 2.4) with a batch size of 32.

အဆင့် (၁) တွင် training data များကို အသုံးပြု၍ အဆင့် (၂) တွင် တည်ဆောက်ခဲ့သည် neural network model ကို train မည်ဖြစ်သည်။ ထိုသို့ train ရာတွင် ပြုလုပ်မည့် အကြိမ် အရေအတွက်ကို ၁၀ ဟု သတ်မှတ်ထားသည်။ ထိုအပြင် dataset ရှိ data ပေါင်း တစ်သိန်းကျော်ကို တပြုပြင်နိုင်တယ် အစား တကြိမ်တွင် ဒေတာ (တိုက်ခန်းတစ်ခန်း၏ အရောင်း အဝယ် မှတ်တမ်း) - ၃၂ ခုကိုသာ အသုံးပြု၍ အကြိမ်ကြိမ် train ပြုလုပ်မည် ဖြစ်သည်။ ထိုသို့ ပြုလုပ်ခြင်းဖြင့် လိုအပ်သည့် memory နှင့် computational load များကို လျော့ချိန်သည်။

```
# =====#
model = create_regression_model(input_shape=[X_train.shape[1]])
model.fit(X_train, y_train, epochs=10, batch_size=32,
           validation_data=(X_test, y_test))
# =====#
```

**☞ Remark**

Batch size refers to the number of training examples utilized in one iteration. Instead of using the entire dataset at once, training is divided into smaller batches. Using batch training allows for more efficient computation, as it reduces the memory requirements and computational load compared to processing the entire dataset at once.

**Step 2.5.1.5**

In the last step, the model's performance is validated using the testing data ( $X_{test}$  and  $y_{test}$ ) and saves the evaluation results to a CSV file.

နောက်ဆုံးအဆင့်တွင် model ၏ performance ကို testing ဒေတာများ အသုံးပြု၍ ဆန်းစစ်မည်။

```
# Evaluate the model
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
from sklearn.metrics import r2_score

df_results = pd.DataFrame(columns=['Train', 'Test'])

y_pred = model.predict(X_train)
df_results.loc['Mean Squared Error', 'Train'] = mean_squared_error(y_train, y_pred)
df_results.loc['Mean Absolute Error', 'Train'] = mean_absolute_error(y_train, y_pred)
df_results.loc['Mean Absolute Percentage Error', 'Train'] =
    mean_absolute_percentage_error(y_train, y_pred)*100
df_results.loc['R2 score', 'Train'] = r2_score(y_train, y_pred)

y_pred = model.predict(X_test)
df_results.loc['Mean Squared Error', 'Test'] = mean_squared_error(y_test, y_pred)
df_results.loc['Mean Absolute Error', 'Test'] = mean_absolute_error(y_test, y_pred)
df_results.loc['Mean Absolute Percentage Error', 'Test'] =
    mean_absolute_percentage_error(y_test, y_pred)*100
df_results.loc['R2 score', 'Test'] = r2_score(y_test, y_pred)

df_results = df_results.round(2)
df_results.to_csv('model_evaluation.csv')
```

## Results and Discussion

In this project, the model was trained without hyper-parameter tuning and subsequently evaluated on unseen test data to assess its efficacy. As illustrated in Table 2.3, the model demonstrates robust predictive capability, with both the training and testing phases yielding results that closely align with actual values.

The similarity in performance between the training and unseen data suggests that the model is not overfitting and is effectively generalizing to new instances. However, it's worth noting that an error of approximately SG\$65,000 may be deemed significant, particularly considering the average price of transactions in the training data, which is reported at SG\$611,563.8. This discrepancy represents roughly a 10% deviation from the actual price, indicating room for further improvement in the model's precision.

ယခု သင်ခန်းစာတွင် ဖော်ပြထားသည့် model သည် hyper-parameter tuning မပြုလုပ်ဘဲ တည်ဆောက်ထားသည့် model ဖြစ်သည်။ သို့သော် လက်ရှိ model၏ ရလဒ်မှာ သင့်တော် ကောင်းမွန်သည်ကို တွေ့ရသည်။ ခန့်မှန်း စွေးနှုန်းနှင့် မူလ စွေးနှုန်း၏ ပျမ်းမျှ ကွာခြားချက်သည် ခြောက် သောင်း ငါးထောင် ရှိပြီး တိုက်ခန်း များ၏ ပျမ်းမျှ ရောင်းစွေးနှုန်းနှင့် နှိုင်းယဉ်ကြည့်ပါက ၁၀ ရာခိုင်နှုန်းလောက်သာ ကွာဟာမှု ရှိသည်။

model ၏ hyper-parameter များဖြစ်သည့် hider layer များနှင့် neuron အရေအတွက် များကို တိုးမြှင့် အသုံးပြုမည်ဆိုပါက ပိုမိုကောင်းမွန်သည့် ရလဒ်များကို ရရှိနိုင်သည်။

Table 2.2. Model Evaluation Metrics

	Train	Test
Mean Squared Error	SG\$86,394.48	SG\$86,284.83
Mean Absolute Error	SG\$65,017.51	SG\$64,877.22
Mean Absolute Percentage Error	10.82 %	10.82 %
R2 score	82 %	82%

### ☞ Remark

The performance can be further enhanced by tuning the hyperparameters, for instance, increasing the size of the hidden layers and depth of the neural network.

## 2.5.2 Multi-class Classification: Name Classification for Regional Identification

The objective of this project is to develop a machine learning model (neural network) specifically tailored to classify names to the corresponding states and regions within Myanmar. This model aims to address the challenges faced in demographic analysis and information extraction from news articles, where non-standardized writing of location names impedes accurate data interpretation.

အများသိကြသည့်အတိုင်းမြန်မာနိုင်ငံတွင် မူလက တိုင်းနှင့်ပြည်နယ် ၁၄ ခု ရှိခဲ့ပြီး ယခုအခါ နေပြည်တော် တိုးလာသည့်အတွက် ၁၅ ခု ရှိသည်။ ထိုအပြင် ရှစ်းကို ရှစ်း တောင်၊ ရှစ်းမြောက်နှင့် ရှစ်း ရွှေ့ဟူ၍ လည်းကောင်း၊ ပဲခူးကို ပဲခူးအနောက်နှင့် ပဲခူး အရှေ့ဟူ လည်းကောင်း ထပ်မံခွဲခြားထားရာ တိုင်းနှင့် ပြည်နယ် ၁၈ ခု ရှိသည်ကို ဆိုရမည်။ ဤ project ၏ ရည်ရွယ်ချက်မှာ နေရာတစ်ခု၏ အမည်ကို ပေးလိုက်သည့်နှင့် မည်သည့် တိုင်းနှင့် ပြည်နယ်အတွင်း ရှိသည်ကို ခန့်မှန်းပေးရန် ဖြစ်သည်။

### Background

In Myanmar, conducting social study research often involves collecting addresses from users, which can be challenging due to the lack of standardization in writing town and region names, particularly for smaller towns. Similarly, in news articles written by both local and international organizations, names are often written in non-standard ways, leading to inaccuracies in extracting location-based information. This project seeks to alleviate these challenges by automating the process of identifying the geographic origins associated with individual names.

မြန်မာ သတင်းအချက်အလက် စီမံခန့်ခွဲရေးအဖွဲ့ (MIMU) [?] ၏ ၂၀၁၉ ခုနှစ် ဒီဇင်ဘာလ စာရင်းအရ မြန်မာနိုင်ငံတွင် မြို့ပေါင်း ၄၆၆ မြို့နှင့် ကျေးရွာပေါင်း ၁၃,၅၉၀ ရှိသည်။ အဆိုပါ ကျေးရွာ အပုံစုံများ၏ အမည် များ နှင့် မြို့အမည် များကို စာလုံးပေါင်းဖော်ပြရာတွင် ဒေသအခေါ်အဝေါ်များနှင့် ကွဲလွှဲမှုများ ရှိနေပါသည်။ ထိုအပြင် ထုတ်ပြန်ထားသည့် အမည်စာရင်းနှင့် ဒေသအခြေပြုသတင်းမီဒီယာများ၏ ဖော်ပြ ချက်များ အကြား ကွဲလွှဲမှုများ ရှိနေပါသည်။ အဆိုပါကွဲလွှဲမှုများကြောင့် လူမှုရေးနှင့် ဆိုင်သည့် သုတေသန လုပ်ငန်းများသည် နေရာ ဒေသအလိုက် အချက်အလက်များကို သုံးသပ်ရာတွင် တိကျမှု ရရှိနိုင်ရန် အကောက်အခဲ များနှင့် ရင်ဆိုင်ကြံတွေ့နေရသည်။ နေရာတစ်ခု၏ အမည်ကို ပေးလိုက်သည့်နှင့် မည်သည့် တိုင်းနှင့် ပြည်နယ်အတွင်း ရှိသည်ကို အလွယ်တကူ ခန့်မှန်းပေးနိုင်မည့် Machine Learning Model တစ်ခု ကို တည်ဆောက်ထားခြင်းဖြင့် ပို၍ တိကျသည့် အချက်အလက်များကို သုံးသပ်ရန် ရည်ရွယ်သည်။

### Dataset

The initial datasets are obtained from the Myanmar Information Management Unit (MIMU) Resource Centre [?], which offers comprehensive lists of names alongside their corresponding states and regions within Myanmar. The first dataset comprises 14,047 villages spread across 18 regions and states, while the second dataset includes 536 towns with their respective regions and states. These datasets serve as references for geographical information.

Additionally, supplementary data is collected from news articles, where variations in the spelling and formatting of town names are common. This dataset consists of 58,789 records spanning the same 18 regions and states. It's important to note that due to the nature of data collection from news articles, duplicate entries are present. Towns and regions are frequently mentioned in the articles, resulting in duplicated data instances.

Each dataset consists of two columns: 'state-region-names' and 'names of the location' (either villages or towns). Subsequently, these three datasets undergo comprehensive cleaning and merging processes to ensure coherence and consistency in the final combined dataset.

ဤ project တွင် အသုံးပြုမည့် dataset ကို မြန်မာ သတင်းအချက်အလက် စီမံခန့်ခွဲရေးအဖွဲ့ (MIMU) မှ ထုတ်နှုတ် ရယူထားခြင်းဖြစ်သည်။ အဆိုပါ အဖွဲ့သည် မြန်မာနိုင်ငံရှိ ဝန်ကြီးဌာနအချိန် အဖွဲ့အစည်းများ၏ အချက်အလက်ကို စုစုပေါင်းတော်ပြထားသည့် ယုံကြည်စိတ်ချရသည့် အဖွဲ့အစည်းတစ်ခု ဖြစ်သည်။ ပထာက် dataset တွင် ကျေးရွာပေါင်း တစ်သောင်း လေးထား လေးဆယ့် ခု ၏ အမည်များနှင့် အဆိုပါ မြို့၊ တည်ရှိသည့် တိုင်းနှင့် ပြည်နယ်၏ အမည်တို့ ပါဝင်သည်။ ခုတိယာ dataset တွင်မူ မြို့ပေါင်း ၅၃၆ မြို့၏ အမည်များနှင့် သက်ဆိုင်ရာ တိုင်းနှင့် ပြည်နယ်၏ အမည်များ ပါဝင်သည်။ ထိုအပြင် သတင်းခေါင်းစွဲများမှာ မြို့ရွာများ၏ အမည်များကို ထုတ်နှုတ်ရယူထားသည့် တတိယာ dataset တွင် ငါးသောင်း ရှစ်ထောင်း ခုနှစ်ရာ ရှစ်ဆယ့်ကိုးခု ပါဝင်သည်။ အထူးသတိပြုရမည့်

အချက်မှာ ဤ တတိယ datasetသည် သတင်းများမှ အချက်အလက်များကို ထုတ်နှစ် စုစဉ်သည် ဖြစ်ရာ အချို့မြို့များ၏ အမည်မှာ တစ်ကြိမ်မက ပါဝင်နိုင်သည်။

dataset ၃ ခု လုံးတွင် column နှစ်ခု သာ ပါဝင်ပြီး ပထမ column တွင် တိုင်းနှင့် ပြည်နယ်၏ အမည်များ ပါဝင်သည်။ ဒုတိယ column တွင်မူ ကျေးရွာ(သို့မဟုတ်) မြို့၏ အမည်များ ပါဝင်သည်။

## Data Preprocessing: Data Cleaning

One of the major challenges in this project is name consistency, even for the names of the regions and states. For instance, the regions and states in the MIMU dataset are defined as "shan (south)," but in the news articles, they are usually defined as "shan-south." This inconsistency necessitates intensive data cleaning to ensure uniformity across datasets.

- ✿ **Standardizing Column Names:** The column names of three datasets are standardized to ensure consistency. Specifically, the columns representing the names of regions and states are renamed to 'SR\_Name' for uniformity across datasets.
- ✿ **Standardizing Naming Convention:** Fuzzy string matching techniques are employed to address variations in naming conventions between datasets. Specifically, the names in the news articles dataset are compared to the standardized names in the MIMU dataset, and the closest matches are identified and used to replace inconsistent names.
- ✿ **Removing Duplicates:** Duplicate rows in the news articles dataset are removed to enhance data quality and reduce redundancy.
- ✿ **Merging** The cleaned and standardized datasets are combined to form a comprehensive dataset (df\_mimu), consolidating information from both the MIMU dataset and news articles. This merged dataset encompasses 19,513 entries across 2 columns, enabling seamless progression to further analysis and modeling.

မြန်မာ သတင်းအချက်အလက် စီမံခန့်ခွဲရေးအဖွဲ့ (MIMU) မှ ရယူထားသည့် dataset တွင် ရှမ်း (တောင်)၊ ရှမ်း (မြောက်) - စသည်ဖြင့် တောင်နှင့် မြောက်ကို ကွင်းစ ကွင်းပိတ်နှင့် အသုံးပြု ဖော်ပြထားသော်လည်း သတင်းဌာနမှ ရယူထားသော ပြည်နယ် အမည်များတွင်မူ ရှမ်း -တောင် ရှမ်း -မြောက် စသည်ဖြင့် ' ' ကို အသုံးပြု ထားသည်။ အဆိုပါ အမည် သတ်မှတ်ချက်များကို ကိုက်ညီစေရန် အတွက် Fuzzy string matching ကို အသုံးပြုသည်။ ထိုအပြင် သတင်းဌာနမှ ရယူထားသည့် dataset ထပ်နေသည့် မြို့အမည်၊ ကျေးရွာ အမည်များကို ဖယ်ထုတ်သည်။ ထိုနောက် dataset ၃ ခု လုံးကို တစ်စုတည်းပေါင်းစည်းလိုက်ပြီး စုပေါင်း dataset တွင် record စုစုပေါင်း - တစ်သောင်း ကိုထောင် ပါးရာ တစ်ဆယ့်သုံးခုနှင့် column နှစ်ခု ပါဝင်သည်။

## Data Preprocessing: Text Encoding (Vectorization)

In this step, the text inputs from the 'name' column and the class labels from the 'SR\_Name' column are transformed into numerical representations. This transformation is necessary to convert the textual data into a format that can be processed by a neural network.

- ✿ Label encoding is applied on the 'SR\_Name' column to map each unique stage/ region name to a numerical value. Label encoding assigns a unique integer label to each unique category in the column.
- ✿ One-hot encoding is applied to the 'name' column because each name represents a specific location, and there is no ordinal relationship between the different names. This encoding method transforms categorical variables into binary vectors, where each unique category (name) is represented by a binary feature. This results in a sparse matrix with 13,003 binary columns, each representing a distinct name.

ဤ dataset တွင် ပါဝင်သည့် အချက်အလက်များမှာ စာသားများဖြင့် ဖော်ပြပ ထားသည် ဖြစ်ရာ ကိန်းကဏ္ဍားများ အဖြစ် ဗိုးစွာ ပြောင်းလဲမည်။ တိုင်းနှင့်ပြည်နယ်များ၏ အမည်များပါဝင်သော 'SR\_Name' column ကို Label encoding အသုံးပြု၍ ကိန်းကဏ္ဍားများအဖြစ် တိုက်ရိုက် ပြောင်းလဲမည်။ ဥပမာ - ရှမ်း (တောင်) ကို ၁၅၊ ရနိုင်ကို ၁၂၊ ဧရာဝတီကို ၇၉ စသည်ဖြင့် သတ်မှတ်သည်။ ထိုနောက် ကျေးရွာအပ်စုများနှင့် မြို့များ၏ အမည်ကို One-hot encoding နည်းစနစ်ကို အသုံးပြု၍ Binary Column များအဖြစ်

# Model Development: Build a Neural Network using Keras

With the merged dataset containing 19,513 entries across 2 columns, the second column 'name' of the location is used as an input feature and the first column 'SR\_Name' is the class label (output) for the model. The section explains the steps for developing a neural network model using Keras for classifying the text inputs to 18 different regions and stages.

## **Step 2.5.2.1**

The first step in our data processing pipeline involves loading the data into a pandas DataFrame. This is accomplished using the pandas library, which is imported at the beginning of the script.

အဆင့် (၁) တွင် ဦးစွာ ပထမ dataset ကို Pandas DataFrame `df` အဖြစ် သိမ်းဆည်းခြင်းဖြစ်သည်။

```
# =====#
# Step 1: Load the data
import pandas as pd
df = pd.read_csv('./data/MMNames_clean.csv')
# =====#
```



Ensure you have the necessary libraries installed. You can install them using pip if they are not already installed. For example:

```
pip install tensorflow pandas scikit-learn
```

သတိပြုရမည့် အချက်မှာ လိုအပ်သည့် library များကို install ပြုလုပ်ရန် ဖြစ်သည်။ အကယ်၍ install မပြုလုပ်ရသေးပါက အထက်ပါ ကုဒ်ကို အသုံးပြခြင်း install ပြုလုပ်ပေးရမည်။

## **Step 2.5.2.2**

This step prepares the data for building the machine learning model. This step involves preprocessing the data to get it ready for training a machine learning model. The code uses custom preprocessing functions in user-defined module 'data\_preprocessing' and utilities from the sklearn library to achieve this.

- The first three lines of the code below import the user-defined module 'data\_preprocessing' and utilities from the sklearn library.
  - The code 'dp.preprocess\_category(df, 'SR\_Name')' preprocesses the 'SR\_Name' column of the DataFrame df using label encoding. This method converts categorical values to numerical values, facilitating their use in machine learning algorithms.
  - The code 'dp.preprocess\_category(df, 'name')' encodes the name column of the DataFrame df into a binary matrix representation using one-hot encoding. This technique enhances the model's ability to interpret and utilize categorical data effectively.
  - The preprocessed data is then split into training and testing sets, with 70% of the data used for training ( $X_{train}$  and  $y_{train}$ ) and 30% for testing ( $X_{test}$  and  $y_{test}$ ). The `random_state=42` ensures reproducibility of the split.

ထိုနောက် 'dp.preprocess\_category(df, 'SR\_Name')' ကုဒ်သည် 'SR\_Name' column ကို Label encoding အသုံးပြု၍ ကိန်းဂဏ်းများအဖြစ် တိုက်ရှိက် ပြောင်းလဲခြင်းဖြစ်ပြီး 'dp.preprocess\_category(df, 'name')'သည် ကျေးမှုအုပ်စုများနှင့် မြို့များ၏ အမည် ကို One-hot encoding နည်းစနစ် ကို အသုံးပြု၍ Binary Column များအဖြစ် ပြောင်းလဲပေးသည်။ ထိုနောက် datasetကို training အတွက် ၇၀ ရာခိုင်နှုန်းကို အသုံးပြုပြီး ကျွန် ၃၀ ရာခိုင်နှုန်းကို testing အတွက် ချန်ထားမည်ဖြစ်သည်။

```
# =====#
# Step 2: Prepare the data
import data_preprocessing as dp
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

df = dp.preprocess_category(df, 'SR_Name')
df = dp.preprocess_onehot(df, 'name')

y = df['SR_Name'].values
X = df.drop(columns=['SR_Name']).values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print(X_train.shape, X_test.shape)
# =====#
```

### Step 2.5.2.3

This step defines a function `create_regression_model` that constructs a feedforward neural network model for classification using TensorFlow's Keras API.

တတိယ အဆင့်သည် TensorFlow ၏ Keras API ကို အသုံးပြု၍ neural network model တစ်ခု တည်ဆောက် ခြင်းဖြစ်သည်။ ယခု model တွင် hidden layer - ၂ ခုသာ ပါဝင်မည်ဖြစ်ပြီး ပထမ hidden layer တွင် neuron - ၃၂ ခု နှင့် ဒုတိယ hidden layer တွင် neuron - ၁၆ ခု ကို အသုံးပြု မည်။ layer - ၂ ခုလုံးအတွက် ReLU activation function ကို အသုံးပြုထားသည်။

- ✿ This function takes `input_shape` and `num_classes` as parameters, representing the shape of the input data and the number of classes.
- ✿ The model comprises an input layer, two hidden layers, and an output layer:
  - ✿ The input layer serves as the initial receiver of input data, passing it to subsequent layers for processing.
  - ✿ The first hidden layer is a fully connected (dense) layer with 32 units and ReLU activation function, as depicted in Figure 2.7.
  - ✿ The second hidden layer is also fully connected, with 16 units and ReLU activation function.
  - ✿ These hidden layers learn intricate patterns among the features.
  - ✿ The output layer has a single unit, representing the classification prediction. The '`softmax`' activation function is used in the output layer.
- ✿ The code `model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])` configures the neural network model.
- ✿ This project employs the Adam optimizer, an adaptive learning rate optimization algorithm. Adam adjusts the learning rate during training, facilitating faster convergence and improved performance.
- ✿ The loss function '`sparse_categorical_crossentropy`' loss function is utilized, which calculates the

cross-entropy loss between the predicted probabilities and the true labels. This loss function is suitable for multi-class classification tasks where the target labels are integers.

- The `accuracy' metric is selected to evaluate the model's performance. It measures the proportion of correctly classified samples out of the total number of samples, providing a straightforward assessment of the model's predictive accuracy.

```
# =====#
import tensorflow as tf

def create_classification_model(input_shape, num_classes, params={}):
    model = tf.keras.Sequential([
        tf.keras.layers.InputLayer(shape=input_shape),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dense(16, activation='relu'),
        tf.keras.layers.Dense(num_classes, activation='softmax')
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    return model
# =====#
```

### Remark

The size of hidden layers and the number of layers in a neural network are critical hyperparameters that influence the model's capacity, performance, and generalization ability. Finding the optimal configuration often involves a balance between model complexity and simplicity, guided by the specific characteristics of the dataset and the requirements of the task at hand.

Hidden layer အရေအတွက်နှင့် neuron အရေအတွက်တို့မှာ neural network တစ်ခုအတွက် အရေးကြီးသည့် hyperparameter များဖြစ်ပြီး dataset နှင့် ဖြေရှင်းမည့် ပြဿနာပေါ်တွင် မူတည်သည်။

### Step 2.5.2.4

The model is trained using the training data ( $X_{train}$  and  $y_{train}$ ). The training process runs for 10 epochs (Refer definition in Section 2.4) with a batch size of 32.

အဆင့် (၁) တွင် training data များကို အသုံးပြု၍ အဆင့် (၂) တွင် တည်ဆောက်ခဲ့သည့် neural network model ကို train မည်ဖြစ်သည်။ ထိုသို့ train ရာတွင် ပြုလုပ်မည့် အကြိမ် အရေအတွက်ကို ဤ ဟု သတ်မှတ်ထားသည်။ ထိုအပြင် dataset ရှိ data ပေါင်း တစ်သိန်းကျော်ကို တပြု့ပြန်ကဲ train ပြုလုပ်မည့် အစား တကြိမ်တွင် ငဒ တာ - ၃၂ ခကိုသာ အသုံးပြု၍ အကြိမ်ကြိမ် train ပြုလုပ်မည် ဖြစ်သည်။ ထိုသို့ ပြုလုပ်ခြင်းဖြင့် လိုအပ်သည့် memory နှင့် computational load များကို လျော့ချိန်သည်။

```
# =====#
model = create_classification_model(input_shape=[X_train.shape[1]],
                                     num_classes=len(df['SR_Name'].unique()), )
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
                     validation_data=(X_test, y_test), verbose=0)
# =====#
```

## Remark

Batch size refers to the number of training examples utilized in one iteration. Instead of using the entire dataset at once, training is divided into smaller batches. Using batch training allows for more efficient computation, as it reduces the memory requirements and computational load compared to processing the entire dataset at once.

## **Step 2.5.2.5**

In the last step, the model's performance is validated using the testing data ( $X_{\text{test}}$  and  $y_{\text{test}}$ ) and saves the evaluation results to a CSV file. In this step, the 'classification\_report' function from scikit-learn is used to generate a classification report for the actual test labels ( $y_{\text{test}}$ ) and the predicted labels ( $y_{\text{pred}}$ ). ແນວກົງທີ່: ຂອງເກີດຕູ້ຈົດ ມີ performance ຖໍ່ testing ເພື່ອມາວະ ແລ້ວ ປຸ່ງ ຂັ້ນທີ່ແລ້ວ.

# Step 5: Evaluate the model

```
from sklearn.metrics import classification_report

y_pred = model.predict(X_test, batch_size=32, verbose=0)
y_pred = y_pred.argmax(axis=1)
report = classification_report(y_test, y_pred, output_dict=True)
report_df = pd.DataFrame(report).transpose()
report_df.to_csv('./data/cls_report_test.csv', index=False)
print(report_df)

y_pred = model.predict(X_train, batch_size=32, verbose=0)
y_pred = y_pred.argmax(axis=1)
report = classification_report(y_train, y_pred, output_dict=True)
report_df = pd.DataFrame(report).transpose()
report_df.to_csv('./data/cls_report_train.csv', index=False)
```

## Results and Discussion

Table 2.3. Model Evaluation Metrics

	<b>Train</b>	<b>Test</b>
Accuracy	86%	28%
weighted average Precision	87%	45%
weighted average Recall	86%	28%

In the results presented above (Ref. Table ??), it's evident that while the model attained an accuracy of 86% on the training data, it only achieved 28% on the test data. Similarly, the model demonstrated weighted average precision and recall rates of 87% and 86% on the training set, but notably dropped to 45% and 28% on the test set. These findings indicate that the developed neural network model, a fully connected multi-layer perceptron, isn't well-suited for this particular problem. Its overfitting can be attributed to its inability to capture sequential patterns and inherent relationships within textual data, such as location names. Such limitations often lead to poor generalization to new data, particularly when dealing with high-dimensional sparse datasets like one-hot encoded vectors.

This is where people have developed more advanced models such as Recurrent Neural Networks (RNNs). RNNs are designed to recognize sequential patterns and dependencies in data by maintaining a hidden state that captures information from previous time steps. This makes RNNs particularly effective for tasks involving sequential data, such as natural language processing, where understanding the order and context of words (or

characters) is crucial.

In the next chapter, we will delve into the advanced deep learning models, particularly one particular section focusing on Recurrent Neural Networks (RNNs). We will explore their design principles, operational mechanisms, and why they are better suited for handling sequential data in tasks like name classification.

အထက်ပါ ယေားတွင် ဖော်ပြထားသည့် ရလဒ်များကို သုံးသပ်ကြည့်မည်ဆိုပါက ယခု သင်ခန်းစာတွင် တင်ပြထားသည့် neural network model (fully connected multi-layer perceptron) သည် အမည်များကို မှန်မှန်ကန်ကန် ခွဲခြားပေးနိုင်ခြင်း မရှိသည် ကို တွေ့ရသည်။ train ပြလုပ်ထားသည့် data အတွက် ဒေါ်ရာခိုင်နှုန်းလောက်ထိ ကောင်းမွန် တိကျသည့် ရလဒ်များကို ထုတ်ပေး နိုင်သော်လည်း အသစ် data များအတွက်မူ ငွေ့ငွေ့ model ၏ ရလဒ်မှာ ဤ ရာခိုင်နှုန်းအောက် ကျဆင်းသွားသည်။ Text data များကို အမျိုးအစား ခွဲခြားကြရတွင် ပိုမိုကောင်းမွန်သည့် deep neural network (deep learning model) များကို ပညာရှင်များမှ တိုတ် ခဲ့ကြပြီး ဖြစ်သည်။ အများဆုံး အသုံးပြုသည့် model မှာ Recurrent Neural Networks (RNNs) ဖြစ်ပြီး နောက် အခန်းတွင် RNN အပါအဝင် advanced deep learning model များကို ဆွေးနွေးသွားမည်။