**Practical 4**

**Multiclass Classification with feed forward network**

**Aim:** Using feed Forward Network with multiple hidden layers for performing multiclass classification and predicting the class.

**Description:**

**Describe multiclass classification in detail.**

* Objective:
  + Multiclass classification aims to assign input data points to one of several predefined classes or categories.
* Dataset:
  + A dataset for multiclass classification consists of input features (independent variables) and corresponding class labels (dependent variables).
  + Each data point in the dataset is associated with a single class label from a set of multiple classes.
* Classes:
  + Classes represent the distinct categories into which data points can be classified.
  + For example, in a dataset of handwritten digit recognition, classes may represent digits from 0 to 9.
* Feature Representation:
  + Input features are represented as a feature vector for each data point.
  + These features could be numerical, categorical, or even text-based, depending on the nature of the problem.
* Model Training:
  + During training, the model learns patterns and relationships between input features and class labels from the labeled training data.
  + Common machine learning algorithms used for multiclass classification include logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks.
* Loss Function:
  + In multiclass classification, a suitable loss function is used to quantify the difference between predicted class probabilities and the true class labels.
  + Common loss functions for multiclass classification include cross-entropy loss, categorical cross-entropy loss, and softmax loss.
* Prediction:
  + After the model is trained, it can be used to predict the class labels for new, unseen data points.
  + The model computes the probability distribution over all classes for each input instance and assigns the class with the highest probability as the predicted class.
* Evaluation Metrics:
  + Evaluation metrics are used to assess the performance of the multiclass classification model.
  + Common evaluation metrics include accuracy, precision, recall, F1-score, confusion matrix, and ROC curves.
* Handling Class Imbalance:
  + In real-world scenarios, class imbalance may occur when some classes have significantly fewer instances than others.
  + Techniques such as oversampling, undersampling, and class-weighted loss functions can be employed to address class imbalance.
* Application:
  + Multiclass classification has various applications across different domains, including image recognition, natural language processing, sentiment analysis, medical diagnosis, and recommendation systems.

**feedforward neural network**  
A feedforward neural network, often referred to as a multilayer perceptron (MLP), is a fundamental architecture in artificial neural networks. It consists of an input layer, one or more hidden layers, and an output layer. Information flows forward through the network, with each layer of neurons processing the input and passing its output to the next layer. Neurons within each layer compute a weighted sum of their inputs, followed by the application of an activation function, which introduces nonlinearity to the network. The network's parameters, including weights and biases, are learned from labeled training data using techniques like gradient descent, enabling the network to map input data to output predictions.

**Code:**

from keras.models import Sequential

from keras.layers import Dense

import pandas as pd

import numpy as np

df = pd.read\_csv("/content/flower\_1.csv")

df.head()

x=df.iloc[:,:-1].astype(float)

y=df.iloc[:,-1]

print(x.shape)

print(y.shape)

#labelencode y

from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

y=lb.fit\_transform(y)

y

import numpy as np

from tensorflow.keras.utils import to\_categorical

#from keras.utils import np\_utils

encoded\_Y = to\_categorical(y)

encoded\_Y

#creating a model

model = Sequential()

model.add(Dense(units = 10, activation = 'relu', input\_dim = 4))

model.add(Dense(units = 8, activation = 'relu'))

model.add(Dense(units = 3, activation = 'softmax'))

model.compile(loss = 'categorical\_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

model.fit(x,encoded\_Y,epochs = 400,batch\_size = 10)

predict = model.predict(x)

print(predict)

for i in range(35,150,3):

print(predict[i],encoded\_Y[i])

actual = []

for i in range(0,150):

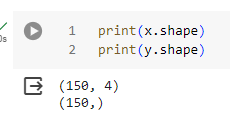
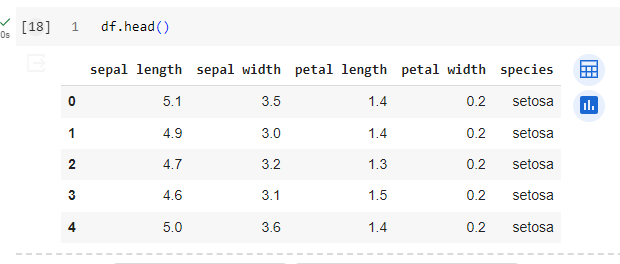
actual.append(np.argmax(predict[i]))

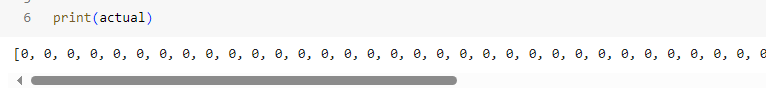
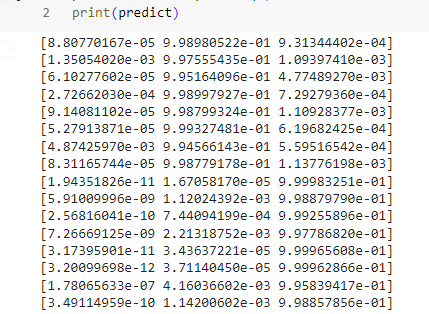
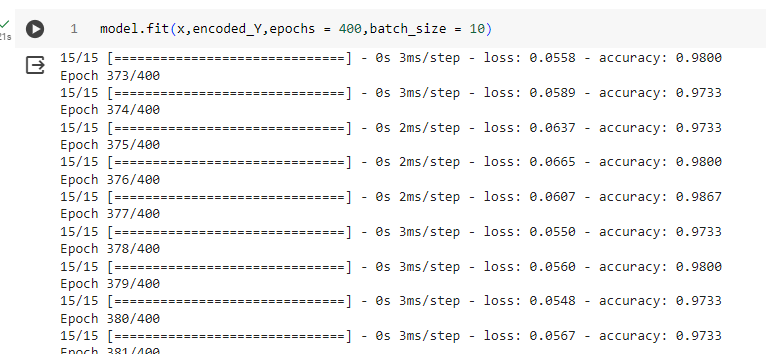
print(actual)

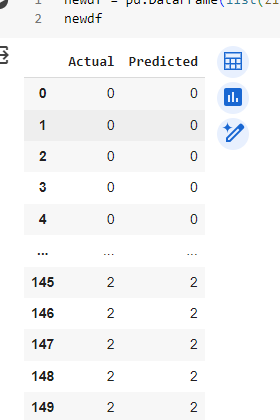
newdf = pd.DataFrame(list(zip(actual,y)),columns = ['Actual','Predicted'])

newdf

**Output:**

****





**Learning:**

This code snippet loads a dataset from a CSV file containing features and labels. It preprocesses the data by encoding the labels and one-hot encoding them. A feedforward neural network is then built using Keras, comprising input, hidden, and output layers. The model is trained using the compiled parameters and training data. After training, it predicts class probabilities for the input data and compares some predictions with actual labels. Finally, the predicted and actual labels are combined into a DataFrame for analysis and evaluation.