**EMPLOYEE CHURN PREDICTION:**

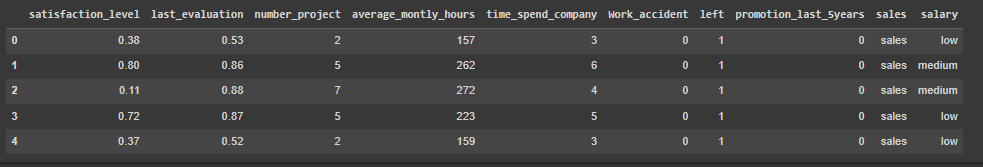
import numpy as np

import pandas as pd

**# Load data**

data=pd.read\_csv('HR\_comma\_sep.csv')

data.head()

****

# Import LabelEncoder

from sklearn import preprocessing

# Creating labelEncoder

le = preprocessing.LabelEncoder()

# Converting string labels into numbers.

data['salary']=le.fit\_transform(data['salary'])

data['sales ']=le.fit\_transform(data['sales '])

# Spliting data into Feature and

X=data[['satisfaction\_level', 'last\_evaluation', 'number\_project', 'average\_montly\_hours', 'time\_spend\_company', 'Work\_accident', 'promotion\_last\_5years', 'sales ', 'salary']]

y=data['left']

# Import train\_test\_split function

from sklearn.model\_selection import train\_test\_split

# Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Import MLPClassifer

from sklearn.neural\_network import MLPClassifier

# Create model object

clf = MLPClassifier(hidden\_layer\_sizes=(6,5),

random\_state=5,

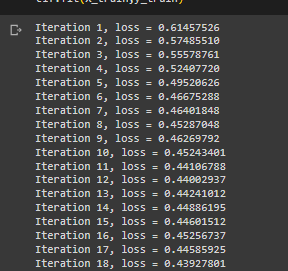
verbose=True,

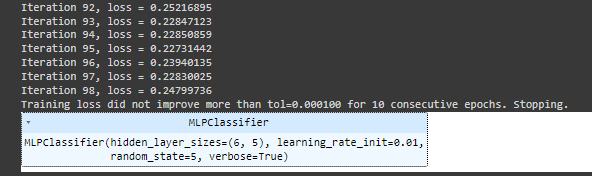
learning\_rate\_init=0.01)

# Fit data onto the model

clf.fit(X\_train,y\_train)

**output:**

****

****

# Make prediction on test dataset

ypred=clf.predict(X\_test)

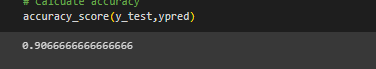
# Import accuracy score

from sklearn.metrics import accuracy\_score

# Calcuate accuracy

accuracy\_score(y\_test,ypred)

output:

****

**Week - 5**

**Module Name: Removing noise from the images**

**Exercise: Implement Multi-Layer Perceptron algorithm for Image denoising hyperparameter tuning**

**#tensorflow week -5**

import tensorflow as tf

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.utils import to\_categorical

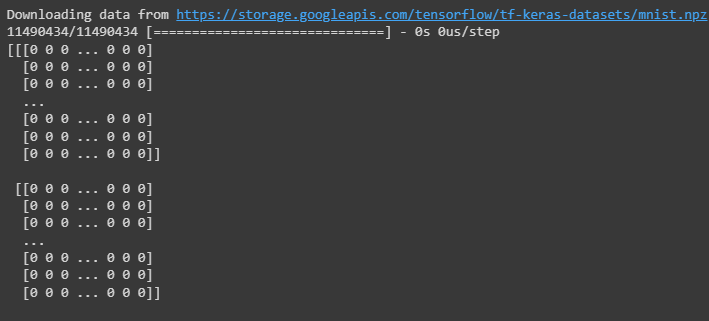
**#load the mnist dataset**

(x\_train,y\_train),(x\_test,y\_test)=mnist.load\_data()

print(x\_train)

**#normalize pixel values to eb between 0 and 1**

x\_train,x\_test=x\_train/255.0,x\_test/255.0

****

y\_train=to\_categorical(y\_train,num\_classes=10)

y\_test=to\_categorical(y\_test,num\_classes=10)

model=Sequential([

Flatten(input\_shape=(28,28)),

Dense(128,activation='relu'),

Dense(64,activation='relu'),

Dense(10,activation='softmax')

])

**#compilethe model**

model.compile(optimizer='adam',loss='categorical\_crossentropy',

metrics=['accuracy'])

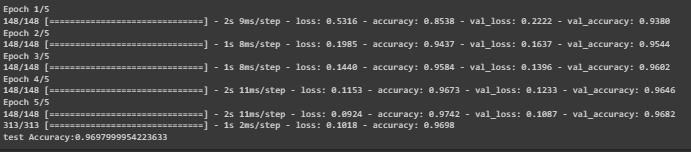
**#train th model**

model.fit(x\_train,y\_train,epochs=5,batch\_size=325,validation\_split=0.2)

**#evaluate the model on the test data**

test\_loss,test\_accuracy=model.evaluate(x\_test,y\_test)

print(f"test Accuracy:{ test\_accuracy}")

****

**Week -1:**

**Course name : .Build a Convolution Neural Network for Image Recognition.**

**Program:**

import numpy as np

import tensorflow as tf

from tensorflow.keras import layers,models

from tensorflow.keras.datasets import mnist

(train\_images,train\_labels),(test\_images,test\_labels)=mnist.load\_data()

**Output:**

****

train\_images,test\_images=train\_images/255.0,test\_images/255.0

train\_labels=tf.keras.utils.to\_categorical(train\_labels,10)

test\_labels=tf.keras.utils.to\_categorical(test\_labels,10)

**# building the cnn**

model=models.Sequential()

model.add(layers.Conv2D(32,(3,3),activation='relu',input\_shape=(28,28,1)))

model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Conv2D(64,(3,3),activation='relu'))

model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Flatten())

model.add(layers.Dense(64,activation='relu'))

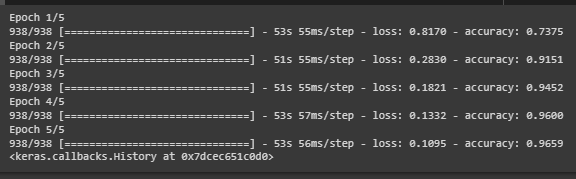
model.add(layers.Dense(10,activation='softmax'))

**#compile the model**

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

**#train the model on training data**

model.fit(train\_images.reshape(-1,28,28,1),train\_labels,epochs=5,batch\_size=64)

****

**#evaluate the model on test data**

test\_loss,test\_accuracy=model.evaluate(test\_images.reshape(-1,28,28,1),test\_labels)

print("test accuracy :",test\_accuracy)

****

**Week-2:**

**Module name : Understanding and Using ANN : Identifying age group of an actor**

**Exercise : Design Artificial Neural Networks for Identifying and Classifying an actor using Kaggle Dataset.**

**Program:**

from tensorflow.keras.models import load\_model

from PIL import Image

import numpy as np

image\_height=128

image\_width=128

num\_channels=3

model=load\_model('trained\_model\_NEW\_2\_2\_Dataset.h5')

new\_face\_path='/content/jk.jpg'

new\_face=image.open(new\_face\_path)

**output:**