Ex No: 2 **BUILD A NEURAL NETWORK WITH KERAS**
Date: 09/08/2024

Aim:

To build a simple neural network using Keras/TensorFlow.

Procedure:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple convolutional neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

Program:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.model_selection import train_test_split

import warnings

warnings.filterwarnings('ignore')

from sklearn import metrics

from sklearn.metrics import confusion_matrix

from sklearn.preprocessing import LabelEncoder
```

```
df=pd.read_csv('iris.csv')

df['species'].value_counts()
ec=LabelEncoder()
df['class1']=ec.fit_transform(df['species'])

df['class1'].value_counts()
df.drop('species',axis=1,inplace=True)

y=df['class1']
X=df.drop('class1',axis=1)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1)

model=Sequential()
model.add(Dense(32,activation='relu',input_shape=(X_train.shape[1],)))
model.add(Dense(64,activation='relu'))
model.add(Dense(1,activation='linear'))

optimizer=tf.keras.optimizers.RMSprop(0.001)
model.compile(loss='mse',optimizer=optimizer,metrics=['mse'])

model.summary()

history=model.fit(X_train,y_train,epochs=10,validation_split=0.2,verbose=1)
```

Output:

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 32)	160
dense_7 (Dense)	(None, 64)	2,112
dense_8 (Dense)	(None, 1)	65

```
Total params: 2,337 (9.13 KB)
Trainable params: 2,337 (9.13 KB)
Non-trainable params: 0 (0.00 B)
Epoch 1/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 97ms/step - loss: 13.2325 - mse: 13.2325 - val_loss: 7.6386 - val_mse: 7.6386
Epoch 2/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 6.6775 - mse: 6.6775 - val_loss: 4.4911 - val_mse: 4.4911
Epoch 3/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 4.1725 - mse: 4.1725 - val_loss: 2.7176 - val_mse: 2.7176
Epoch 4/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - loss: 2.4993 - mse: 2.4993 - val_loss: 1.6668 - val_mse: 1.6668
Epoch 5/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 1.5960 - mse: 1.5960 - val_loss: 1.0374 - val_mse: 1.0374
Epoch 6/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 1.0600 - mse: 1.0600 - val_loss: 0.6674 - val_mse: 0.6674
Epoch 7/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.6534 - mse: 0.6534 - val_loss: 0.4277 - val_mse: 0.4277
Epoch 8/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.4946 - mse: 0.4946 - val_loss: 0.2980 - val_mse: 0.2980
Epoch 9/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.3227 - mse: 0.3227 - val_loss: 0.2137 - val_mse: 0.2137
Epoch 10/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.2322 - mse: 0.2322 - val_loss: 0.1626 - val_mse: 0.1626
```

Result:

Thus the program to build a simple neural network using Keras/TensorFlow is implemented successfully.