EX NO: 02

DATE:

MULTI LAYER PERCEPTRON WITH HYPER PARAMETER TUNNING

AIM:

To build a Multi-Layer Perceptron (MLP) model for liver disease classification and evaluate its accuracy after preprocessing and training.

ALGORITHM:

Step 1: Import required libraries such as pandas, numpy, sklearn, tensorflow, and keras-tuner.

Step 2: Load the dataset (indian_liver_patient.csv) using pandas.

Step 3: Display the first few records using df.head() to understand the data structure.

Step 4: Rename the target column Dataset to Label.

Step 5: Convert the target values — 1 for disease and 0 for no disease.

Step 6: Remove rows with missing values using df.dropna().

Step 7: Convert the Gender column to numerical form: 1 for Male, 0 for Female.

Step 8: Split the features (X) and the label (y).

Step 9: Apply StandardScaler to normalize the feature values.

Step 10: Split the dataset into training and testing sets using train_test_split.

Step 11: Create a Sequential MLP model with Dense and Dropout layers.

Step 12: Compile the model using the Adam optimizer and binary crossentropy loss.

Step 13: Train the model using the training set with validation split and batch size.

Step 14: Predict the output on the test data and round the predictions to 0 or 1.

Step 15: Evaluate the accuracy using accuracy_score and print the result.

CODING:

```
!pip install keras-tuner

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import keras_tuner as kt

df = pd.read_csv("/content/drive/MyDrive/Deep
Learning/indian_liver_patient.csv")
df.head()


df.rename(columns={'Dataset': 'Label'}, inplace=True)

df['Label'] = df['Label'].apply(lambda x: 1 if x == 1 else 0)

df = df.dropna()

df['Gender'] = df['Gender'].apply(lambda x: 1 if x == 'Male' else 0)

X = df.drop('Label', axis=1)
y = df['Label']

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=32)

model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.summary()
```

```python
history = model.fit(X_train, y_train, epochs=30, batch_size=16,
validation_split=0.2, verbose=1)
y_pred = (model.predict(X_test) > 0.5).astype("int32")

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

OUTPUT:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 64)	704
dropout_3 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 32)	2,080
dropout_4 (Dropout)	(None, 32)	0
dense_6 (Dense)	(None, 1)	33

```
Total params: 2,817 (11.00 KB)
Trainable params: 2,817 (11.00 KB)
Non-trainable params: 0 (0.00 B)
```

```
Epoch 1/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.4970 - loss: 0.6973 - val_accuracy: 0.7419 - val_loss: 0.5838
Epoch 2/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6636 - loss: 0.6184 - val_accuracy: 0.7419 - val_loss: 0.5396
Epoch 3/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6962 - loss: 0.5502 - val_accuracy: 0.7419 - val_loss: 0.5198
Epoch 4/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.6362 - loss: 0.5937 - val_accuracy: 0.7419 - val_loss: 0.5115
Epoch 5/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7067 - loss: 0.5434 - val_accuracy: 0.7419 - val_loss: 0.5030
Epoch 6/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7193 - loss: 0.5275 - val_accuracy: 0.7419 - val_loss: 0.4967
Epoch 7/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7073 - loss: 0.5375 - val_accuracy: 0.7419 - val_loss: 0.4941
Epoch 8/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7217 - loss: 0.5013 - val_accuracy: 0.7419 - val_loss: 0.4886
Epoch 9/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7346 - loss: 0.4813 - val_accuracy: 0.7419 - val_loss: 0.4805
Epoch 10/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7051 - loss: 0.5278 - val_accuracy: 0.7419 - val_loss: 0.4776
Epoch 11/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7546 - loss: 0.4926 - val_accuracy: 0.7419 - val_loss: 0.4785
Epoch 12/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6863 - loss: 0.5367 - val_accuracy: 0.7742 - val_loss: 0.4779
Epoch 13/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.6977 - loss: 0.5042 - val_accuracy: 0.7742 - val_loss: 0.4739
Epoch 14/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7154 - loss: 0.5298 - val_accuracy: 0.7742 - val_loss: 0.4769
Epoch 15/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.7226 - loss: 0.4909 - val_accuracy: 0.7527 - val_loss: 0.4790
```

```
Epoch 16/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7206 - loss: 0.4780 - val_accuracy: 0.7849 - val_loss: 0.4789
Epoch 17/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.7224 - loss: 0.4916 - val_accuracy: 0.7527 - val_loss: 0.4794
Epoch 18/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.6801 - loss: 0.4999 - val_accuracy: 0.7849 - val_loss: 0.4762
Epoch 19/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.6877 - loss: 0.5557 - val_accuracy: 0.7742 - val_loss: 0.4791
Epoch 20/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.7045 - loss: 0.4995 - val_accuracy: 0.7742 - val_loss: 0.4833
Epoch 21/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.6702 - loss: 0.5214 - val_accuracy: 0.7742 - val_loss: 0.4839
Epoch 22/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7033 - loss: 0.4872 - val_accuracy: 0.7527 - val_loss: 0.4819
Epoch 23/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7230 - loss: 0.4848 - val_accuracy: 0.7742 - val_loss: 0.4777
Epoch 24/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7440 - loss: 0.4544 - val_accuracy: 0.7742 - val_loss: 0.4752
Epoch 25/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7341 - loss: 0.4603 - val_accuracy: 0.7742 - val_loss: 0.4760
Epoch 26/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7347 - loss: 0.4787 - val_accuracy: 0.7742 - val_loss: 0.4790
Epoch 27/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7528 - loss: 0.4719 - val_accuracy: 0.7742 - val_loss: 0.4792
Epoch 28/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7303 - loss: 0.4935 - val_accuracy: 0.7634 - val_loss: 0.4832
Epoch 29/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6777 - loss: 0.5142 - val_accuracy: 0.7634 - val_loss: 0.4848
Epoch 30/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7313 - loss: 0.4752 - val_accuracy: 0.7527 - val_loss: 0.4824
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Accuracy: 72.41%
```

COE (20):	
RECORD (20):	
VIVA (10):	
TOTAL (50):	

RESULT:

The MLP model was successfully trained and tested on the liver patient dataset, and its accuracy was evaluated and printed.