```
# Class: B.E. A
# Batch: A1
# Name: Aryan Ghatge
# Roll No.: 4101005
# LP-V (DL) lab-5


import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import fetch_california_housing


# Load and split the dataset (Independent variables X, Dependent variable y)
data = fetch_california_housing()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.2, random_state=42)


# Standardize the data to improve model performance
scaler = StandardScaler()
X_train, X_test = scaler.fit_transform(X_train), scaler.transform(X_test)


# Define the Deep Neural Network (DNN) model with enhancements
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),  # Increased neurons
    Dropout(0.2),  # Dropout to prevent overfitting
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dense(1)  # Linear output layer for predicting continuous values
])
```

```
/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
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# Compile the model with a lower learning rate
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), loss='mse', metrics=['mae'])


# Train the model using Backpropagation with increased epochs and different batch size
model.fit(X_train, y_train, epochs=100, batch_size=16, validation_data=(X_test, y_test))
```

Show hidden output

```
# Evaluate model performance on test data
test_mae = model.evaluate(X_test, y_test)[1]
```

```
129/129 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2508 - mae: 0.3327
```

```
# Make predictions on the test set
y_pred = model.predict(X_test[:5]).flatten()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
```

```
# Display results
print(f"Test MAE: {test_mae}")
print("Actual Prices:", y_test[:5])
print("Predicted Prices:", y_pred)
```

```
Test MAE: 0.32886597514152527
Actual Prices: [0.477   0.458   5.00001 2.186   2.78   ]
Predicted Prices: [0.564749  1.0922694 5.1393127 2.673568  2.475213 ]
```