EX NO:

DATE:

A MULTILAYER PERCEPTRON WITH A HYPERPARAMETER TUNNING

AIM:

To build a Multilayer Perceptron (MLP) model with hyperparameter tuning using Keras Tuner, for predicting student academic performance (pass/fail) based on features.

ALGORITHM:

STEP 1: Import all necessary tools like pandas, Keras, and scikit-learn.

STEP 2: Load your dataset (CSV file) into Python.

STEP 3: Create a new column to mark students as pass or fail based on grades.

STEP 4: Remove unnecessary columns like the original grades.

STEP 5: Scale the input values so all features have similar range.

STEP 6: Split the data into training and testing parts.

STEP 7: Write a function to build the neural network (MLP) model.

STEP 8: Use a tuner to try different model settings (like number of units, layers).

STEP 9: Let the tuner find the best model using the training data.

STEP 10: Get the best model from the tuner.

STEP 11: Train this best model more using your training data.

STEP 12: Test the model using test data and check accuracy.

CODING:

```
import pandas as pd

df = pd.read_csv('/content/data (1).csv')
print(df.columns.tolist())
```

```
['Socioeconomic Score', 'Study Hours', 'Sleep Hours', 'Attendance (%)', 'Grades']
```

```
df.head()
```

Socioeconomic Score	Study Hours	Sleep Hours	Attendance (%)	Grades	pass
0	0.95822	3.4	8.2	53.0	47.0
1	0.85566	3.2	5.9	55.0	35.0
2	0.68025	3.2	9.3	41.0	32.0
3	0.25936	3.2	8.2	47.0	34.0
4	0.60447	3.8	10.0	75.0	33.0

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 1. Load Dataset
df = pd.read_csv('/content/data (1).csv')

# 2. Create binary target
df['pass'] = (df['Grades'] >= 50).astype(int)

# 3. Drop target from features
X = df.drop(['Grades', 'pass'], axis=1)
y = df['pass']

# 4. Normalize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 5. Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# 6. Build MLP model with 3 hidden layers
model = Sequential([
    Dense(64, activation='relu', input_shape=(X.shape[1],)),  # Input layer
    Dense(32, activation='relu'),  # Hidden Layer 1
    Dense(16, activation='relu'),  # Hidden Layer 2
    Dense(1, activation='sigmoid')  # Output Layer
])

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

```
# 7. Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=8, validation_split=0.1,
verbose=0)

# 8. Evaluate
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print(f"\n Final Test Accuracy: {test_acc * 100:.2f}%")

# 9. Classification report
y_pred = (model.predict(X_test) > 0.5).astype(int)
print("\n Classification Report:")
print(classification_report(y_test, y_pred))
```

OUTPUT:

```
Final Test Accuracy: 98.20%
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.99      0.99       230
           1       0.96      0.94      0.95        48

    accuracy                           0.98       278
   macro avg       0.97      0.96      0.97       278
weighted avg       0.98      0.98      0.98       278
```

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

RESULT:

The Multilayer Perceptron (MLP) model was successfully trained and tested on the student academic performance and its accuracy was evaluated and printed.