**Assignment No: - 1**

**Name :- Isha Ghorpade**

**Enrollnment :- 22420020**

**Roll :- 381066**

**Problem Statement:**

Implementing Feedforward neural networks in Python using Keras and TensorFlow.

**Objective**

* To explore the working principle of feedforward neural networks.
* To understand the role of preprocessing in preparing data for neural models.
* To design and implement a feedforward neural network using TensorFlow and Keras.
* To measure model performance using training and validation sets.
* To visualize the learning process through training and validation loss curves.

**Theory**

A Feedforward Neural Network (FNN) is a neural architecture where data flows in one direction — from the input layer to the output layer — without feedback loops. This type of network is considered the foundation of deep learning models.

* Input Layer: Accepts raw input features from the dataset.
* Hidden Layers: Contain neurons that process information using weighted connections and activation functions. Each neuron is connected to the next layer, forming a fully connected structure.
* Output Layer: Generates the final result (e.g., probability, class label, or numerical value).

The network learns by adjusting its weights during training. Activation functions such as ReLU, sigmoid, and softmax are essential to introduce non-linearities, enabling the network to learn more complex mappings between input and output.

The training phase relies on backpropagation, where errors are calculated and propagated backward to update weights using optimization algorithms like Adam. This iterative process minimizes loss, allowing the network to improve predictions over time.

**Methodology**

1. Data Acquisition:
   * Load the winequality-red.csv dataset containing chemical attributes of red wine and its quality score.
2. Data Preprocessing:
   * Divide the dataset into 75% training data and 25% validation data.
   * Normalize feature values to the range (0,1) for faster and more stable training.
3. Model Design:
   * Create a sequential feedforward neural network with Keras.
   * Layers:
     + Dense layer (64 units, ReLU activation) – Input layer.
     + Dense layer (64 units, ReLU activation) – Hidden layer.
     + Dense layer (1 unit) – Output layer for regression.
4. Compilation:
   * Optimizer: Adam
   * Loss Function: Mean Absolute Error (MAE)
5. Training:
   * Train the model on the training set and validate on the validation set across multiple epochs.
6. Evaluation:
   * Use the validation dataset to test predictions and compare them with actual results.
7. Visualization:
   * Plot the training and validation loss values across epochs to identify trends and possible overfitting.

**Advantages**

* Captures Complex Relationships: Can represent non-linear dependencies between inputs and outputs.
* Flexible Structure: The architecture can be adapted to suit regression, classification, or other tasks.
* Scalable: Adding more layers and neurons improves learning capabilities for large datasets.
* Generalization Ability: Properly trained networks perform well on unseen data.
* Fast Processing: Parallel computation allows efficient use of GPUs for quick training.

**Limitations**

* Data Hungry: Requires large amounts of data to perform effectively.
* Expensive Training: High computational resources and training time are often required.
* Opaque Decision-Making: Acts as a "black box," making it hard to interpret results.
* Overfitting Risk: Complex networks may memorize training data instead of generalizing.
* Hyperparameter Sensitivity: Performance strongly depends on tuning of learning rate, number of layers, etc.

**Applications**

* Classification: Email spam detection, sentiment analysis, image recognition.
* Regression: Predicting wine quality, stock market trends, or real estate prices.
* Signal & Speech Processing: Speech recognition, audio classification.
* Healthcare: Predicting disease risks, analyzing medical images.
* Industry: Quality inspection, demand forecasting, and fault detection in machinery.

**Working / Algorithm**

Step 1: Import required libraries (NumPy, Pandas, TensorFlow, Keras, Matplotlib).  
Step 2: Load the dataset (winequality-red.csv).  
Step 3: Split data into training (75%) and validation (25%) sets.  
Step 4: Normalize features using Min-Max scaling.  
Step 5: Separate input features (X) and target variable (y).  
Step 6: Define input shape from training features.  
Step 7: Build the model:

* Dense(64, ReLU) → Dense(64, ReLU) → Dense(1).  
  Step 8: Compile with Adam optimizer and MAE loss function.  
  Step 9: Train the model and monitor validation loss.  
  Step 10: Evaluate predictions on validation set.  
  Step 11: Plot training and validation loss across epochs.  
  Step 12: Use trained model for new predictions.

**Conclusion**

The feedforward neural network is a fundamental deep learning model that can handle both classification and regression tasks. In this experiment, the model was successfully applied to predict wine quality using chemical attributes. The preprocessing, model building, and evaluation steps highlight the importance of data normalization, suitable architecture design, and performance visualization.

Although FNNs face challenges such as high computational requirements and risks of overfitting, they remain highly effective when paired with sufficient data and proper tuning. Their adaptability, scalability, and ability to capture non-linear relationships make them a powerful tool in solving complex real-world problems.