**Assignment No: 1**

**Problem Statement:**

Implementing Feedforward neural networks in Python using Keras and TensorFlow

**Objective:**

* To understand the fundamental architecture of feedforward neural networks.
* To explore data preprocessing techniques for neural network training.
* To design and implement a feedforward network model using Keras with TensorFlow backend.
* To assess the model’s performance using validation datasets.
* To visualize the training and validation loss progression across epochs.

**Software & Hardware Requirements:**

* **Operating System:** Windows/Linux/MacOS
* **Programming Environment:** Python 3.x with Jupyter Notebook, Anaconda, or Google Colab
* **Hardware:** Minimum CPU with 4GB RAM (GPU recommended for faster computation)
* **Libraries/Packages:** TensorFlow, Keras, NumPy, Pandas, Matplotlib, Scikit-Learn

**Theory:**

A **Feedforward Neural Network (FNN)** is a type of artificial neural network where the data flows in one direction—from the input layer to hidden layers (if present), and finally to the output layer—without any feedback loops.

**Key Components:**

* **Input Layer:** Accepts input features from the dataset.
* **Hidden Layers:** Perform transformations using weighted connections. Each neuron is fully connected to the next layer.
* **Output Layer:** Produces the final prediction or classification.

**Important Concepts:**

* **Activation Functions:** Non-linear functions such as ReLU, Sigmoid, or Softmax are applied to introduce complexity and allow the model to capture non-linear relationships.
* **Backpropagation:** The learning algorithm that propagates error backward to update weights and improve model accuracy.

**Methodology:**

1. **Data Loading**
   * Import the dataset (winequality-red.csv) using Pandas to analyze chemical features of wine and their corresponding quality scores.
2. **Data Preprocessing**
   * Split the dataset into **training set (75%)** and **validation set (25%)**.
   * Apply normalization (min-max scaling) to rescale features between 0 and 1 for efficient training.
3. **Model Design**
   * Create a Sequential model in Keras.
   * Add dense layers:
     + Input Layer → 64 neurons with ReLU activation.
     + Hidden Layer → 64 neurons with ReLU activation.
     + Output Layer → 1 neuron for regression output.
4. **Compilation**
   * Use **Adam optimizer**.
   * Set **Mean Absolute Error (MAE)** as the loss function.
5. **Training**
   * Train the model on training data with validation monitoring.
   * Track loss across multiple epochs.
6. **Evaluation**
   * Predict wine quality on validation data.
   * Compare predicted values against actual values.
7. **Visualization**
   * Plot **training loss vs. validation loss** to analyze performance trends and detect overfitting.

**Advantages:**

* **Non-linear Modeling:** Activation functions enable learning of complex relationships.
* **Customizable:** Network structure can be easily adapted (layers, neurons, activations).
* **Scalable:** Works efficiently with larger datasets and deeper architectures.
* **Generalization:** Can achieve strong performance on unseen data if trained correctly.
* **Hardware Optimization:** Supports GPU parallelism for faster computations.

**Limitations:**

* **High Data Requirement:** Needs large labeled datasets for effective training.
* **Resource Intensive:** Training deep models is computationally expensive.
* **Lack of Interpretability:** Often acts as a “black box,” making decisions hard to explain.
* **Overfitting Risk:** May capture noise if the model is too complex.
* **Hyperparameter Sensitivity:** Requires careful tuning of parameters like learning rate, number of layers, etc.

**Applications:**

* Regression problems (e.g., predicting wine quality, housing prices).
* Classification tasks (image, text, or speech recognition).
* Financial forecasting and risk analysis.
* Healthcare predictions and diagnostics.
* Industrial process optimization.

**Working / Algorithm:**

1. Import required libraries (NumPy, Pandas, TensorFlow, Keras).
2. Load the dataset (winequality-red.csv) into Pandas DataFrame.
3. Split dataset → 75% training, 25% validation.
4. Normalize features using Min-Max scaling.
5. Separate input features (X) and output labels (y).
6. Define input shape based on feature size.
7. Build Sequential model:
   * Dense(64, ReLU), Dense(64, ReLU), Dense(1).
8. Compile with Adam optimizer and MAE loss.
9. Train the model with training set and validate on validation set.
10. Evaluate predictions against validation data.
11. Plot training vs. validation loss curves.
12. Model ready for prediction on new data.

**Conclusion:**

The **Feedforward Neural Network (FNN)** is a fundamental yet powerful architecture for supervised learning tasks. By leveraging non-linear activation functions and backpropagation, it can capture intricate relationships in data. Although it requires substantial data, resources, and careful tuning, FNNs remain widely applicable in regression and classification problems. In this experiment, predicting wine quality illustrates how FNNs can successfully transform input features into meaningful predictions, demonstrating both their strengths and practical challenges.