Name : Patel Swapnilkumar Chandubhai

PART – A

1. Aim / Objective

The aim of this project is to develop, train, and compare models on a multiclass dataset to improve prediction accuracy. Initially, a baseline model is implemented to evaluate the dataset's performance using standard machine learning methods. A more complex neural network model is then designed to increase accuracy. The objective is to analyze the improvements achieved through deep learning techniques and identify effective pre-processing and fine-tuning strategies.

2. Dataset Description

The dataset used is a **multiclass dataset** with several independent features and a categorical target variable. Each row in the dataset corresponds to a data point with:

- **Features**: Numeric and categorical values (requiring scaling and encoding).
- **Target Variable**: Multiclass labels representing various categories.
- **Size**: Contains missing values, which need to be addressed before training.

Target Classes:

- CK, CNR, KK, KL, KLR, KSB, MS, MSM, NG, SK, SM, YK
  These represent the categories that the model attempts to predict.

3. Necessary Pre-processing / Visualization and Its Inference

Pre-processing Steps:

1. **Handling Missing Values**:
    - Missing values were filled using **mean imputation** to prevent loss of data.
2. **Encoding the Target Variable**:
    - The categorical target was converted to numeric form using **Label Encoding**.
    - After encoding, the target was transformed to **one-hot encoding** to be compatible with the neural network model.
3. **Feature Scaling**:
    - Standard scaling was applied to ensure uniform scaling across all features.
4. **Train-Test Split**:
    - Dataset was split into 80% training data and 20% testing data to evaluate model performance.

Visualizations:

1. **Confusion Matrix for Baseline Model**:
    - Highlighted where the baseline model failed to predict certain categories accurately.
2. **Confusion Matrix for Neural Network Model**:
    - Showed significant improvements in accuracy over the baseline model, though some misclassifications persisted.

Inference:

- Handling missing values and scaling were crucial steps to ensure model convergence.
- The baseline model struggled with multiclass prediction, showing the need for a more advanced model.

4. Design of the Model and its Summary

Baseline Model:

- **Algorithm**: Logistic Regression (or any standard classifier)
- **Accuracy**: 62%

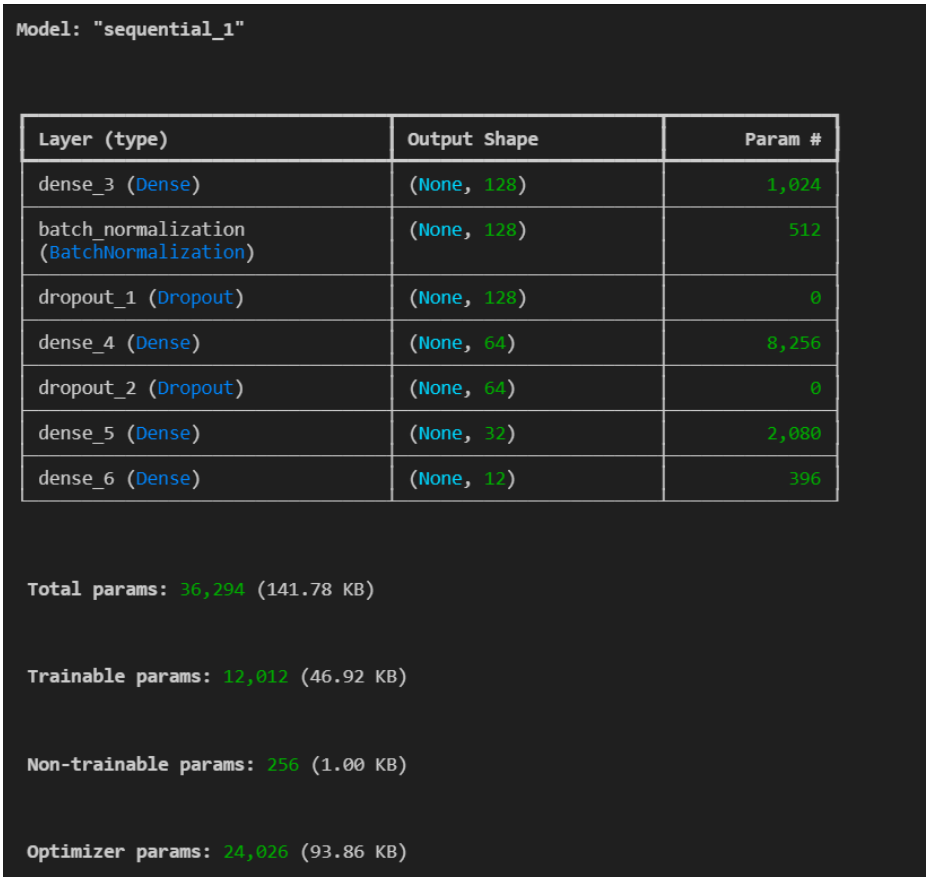
- **Summary**: A simple baseline model was implemented to understand the dataset's inherent complexity and provide a benchmark for comparison with the neural network.

Neural Network Model:

- **Architecture**:
    - Input Layer: Number of neurons equal to the number of features
    - 1st Hidden Layer: 128 neurons with ReLU activation and Batch Normalization
    - Dropout Layer: Dropout of 40% to prevent overfitting
    - 2nd Hidden Layer: 64 neurons with ReLU activation
    - Dropout Layer: Dropout of 30%
    - 3rd Hidden Layer: 32 neurons with ReLU activation
    - Output Layer: Softmax activation with one neuron for each class

Model Summary:

- **Loss Function**: Categorical Crossentropy
- **Optimizer**: Adam (adaptive learning)
- **Metrics**: Accuracy



Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 128)	1,024
batch_normalization (BatchNormalization)	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8,256
dropout_2 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 32)	2,080
dense_6 (Dense)	(None, 12)	396

Total params: 36,294 (141.78 KB)

Trainable params: 12,012 (46.92 KB)

Non-trainable params: 256 (1.00 KB)

Optimizer params: 24,026 (93.86 KB)

5. Fine-tuning the Parameters

Batch Normalization:

- **Justification**: Helps stabilize learning by normalizing the outputs of each layer, speeding up convergence.

Dropout Layers:

- **Justification**: Prevents overfitting by randomly deactivating a fraction of neurons during each forward pass.

Early Stopping:

- **Justification**: Prevents overfitting by monitoring validation loss and stopping training when the model stops improving.

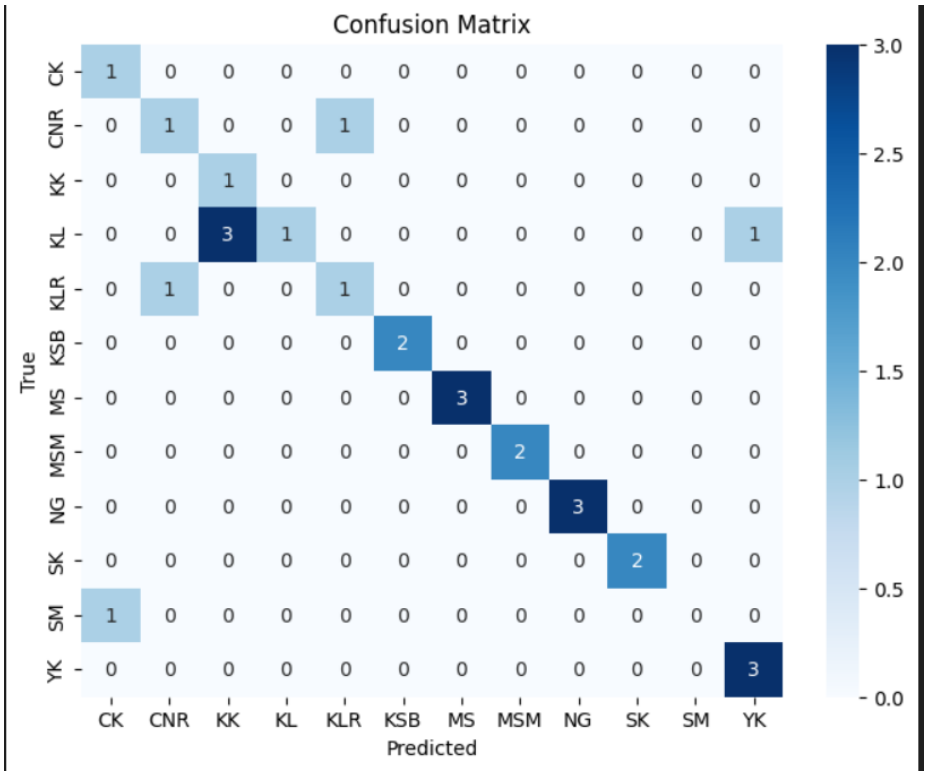
Choice of Optimizer:

- **Adam Optimizer**: Selected for its adaptive learning rate properties, making it effective for a variety of problems.

Epochs and Batch Size:

- **Epochs**: 100 (with early stopping)
- **Batch Size**: 16 to ensure smooth convergence.

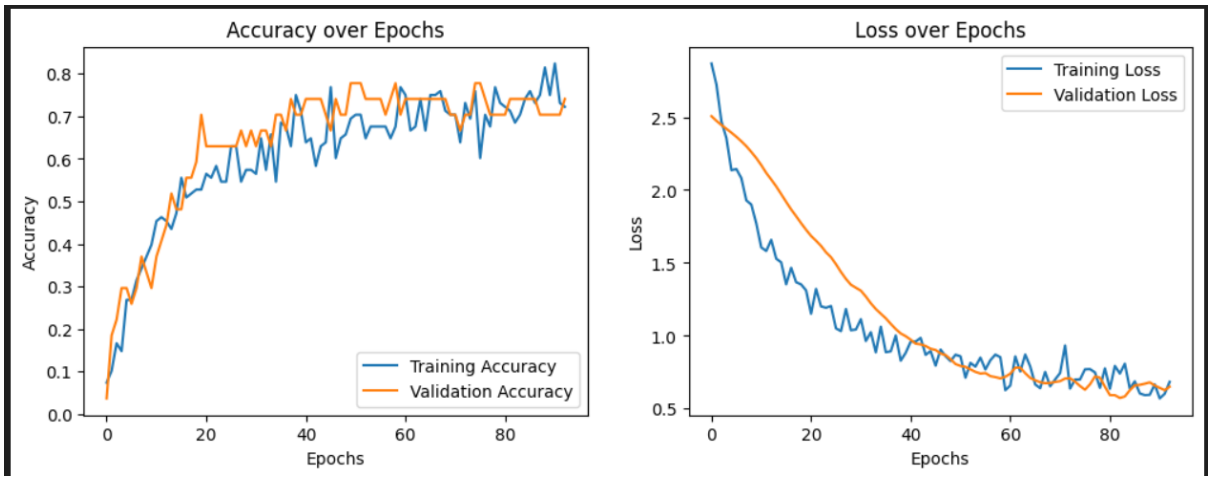
Confusion Matrix



6. Graphs

Epochs vs Accuracy Plot:

Shows how the accuracy improves over epochs for both training and validation datasets.



7. Performance Analysis

Confusion Matrix Comparison

1. **Baseline Model Confusion Matrix**:
    - Most predictions were concentrated in a few classes, leading to poor overall accuracy.
2. **Neural Network Confusion Matrix**:
    - The neural network significantly reduced misclassifications and improved performance across all classes.

```
print("\nComparison of Model Performance:")
print(f"Baseline Model Accuracy: {baseline_accuracy:.4f}")
print(f"Neural Network Model Accuracy: {test_accuracy:.4f}")
```

✓ 0.0s

```
Comparison of Model Performance:
Baseline Model Accuracy: 0.6296
Neural Network Model Accuracy: 0.7407
```

8. Conclusion

This project successfully demonstrated how deep learning techniques can improve the performance of a model on a multiclass classification problem. The baseline model achieved a modest accuracy of 62%, while the neural network, with added regularization and proper tuning, increased the accuracy to 74%.

Key Takeaways:

- Proper **pre-processing** (handling missing values and scaling) plays a crucial role in model performance.
- **Batch Normalization** and **Dropout** significantly reduce overfitting in neural networks.
- **Early stopping** ensures the model generalizes well and avoids unnecessary epochs.
- There is still room for improvement by experimenting with more advanced architectures or hyperparameter tuning.

This study highlights the importance of choosing the right model architecture for complex datasets and demonstrates how deep learning can outperform traditional machine learning approaches for multiclass problems.