Model Optimization and Tuning Phase Template

Date	15 October 2024
Team ID	739739
Project Title	Predicting Diamond Prices With ANN Using Deep Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters

Keras	[ ]
	`import keras_tuner as kt`
	`from tensorflow import keras`
	
	`# Define a function for the model with hyperparameters`
	`def build_model(hp):`
	`  model = keras.Sequential()`
	
	`  # Tune the number of neurons in the first dense layer`
	`  model.add(keras.layers.Dense(`
	`  units=hp.Int('units_layer1', min_value=64, max_value=256, step=32),`
	`  activation='relu',`
	`  input_shape=(x_train.shape[1],)))`
	
	`  # Tune the number of neurons in the second dense layer`
	`  model.add(keras.layers.Dense(`
	`  units=hp.Int('units_layer2', min_value=32, max_value=128, step=16),`
	`  activation='relu'))`

	Hyperparameter tuning optimizes the performance of a machine learning model by testing different combinations of parameters such as the number of neurons, learning rate, and more. The code uses Keras Tuner to automate this process by defining a search space and testing two configurations (limited by max_trials=2). The model includes tunable parameters like the number of neurons in each dense layer and the learning rate. The tuner evaluates these configurations based on validation accuracy or loss, helping find the best combination for your dataset. After tuning, the best parameters are used to build and train the final model. This approach ensures efficient experimentation without manual trial and error.

Final Model	**Reasoning**
Keras	Keras was chosen as the final optimized model for predicting diamond prices due to its simplicity, flexibility, and robust capabilities for building and fine-tuning deep learning models. Keras provides an intuitive interface for designing and training neural networks, making it easier to experiment with different architectures and hyperparameters. Keras integrates seamlessly with TensorFlow, offering powerful tools for scalability, distributed training, and deployment. The model's ability to handle both numeric and categorical features. Its proven performance in regression tasks and real-world applications validates its selection as the final, optimized model for accurately predicting diamond prices.