

## Model Optimization and Tuning Phase Report

Date	15 March 2024
Team ID	SWTID1720188483
Project Title	Nutrition App Using Gemini Pro : Your Comprehensive Guide To Healthy Eating And WellBeing
Maximum Marks	10 Marks

### Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning 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.

### Fine-Tuning Documentation (6 Marks):

In this case we are dealing

Model	Fine Tuning	Optimal Values
Temperature	<pre>def generate_text_with_temperature(prompt, temperature):     """Generates text using the Generative AI model with specified temperature.      Args:         prompt: The text prompt to provide to the model.         temperature: A value between 0 (low randomness) and 1 (high randomness)             to control the creativity of the generated text.      Returns:         The generated text by the model.     """     try:         generated_text = generative_ai.generate(             prompt=prompt,             temperature=temperature         )         return generated_text     except (AttributeError, NotImplementedError):         print("Temperature control not supported by this library/API.")         return None  # Example Usage user_prompt = "Write a story about a robot chef." text_with_low_randomness = generate_text_with_temperature(user_prompt, 0.1) text_with_high_randomness = generate_text_with_temperature(user_prompt, 0.8)  print("Text with low randomness:") print(text_with_low_randomness)  print("Text with high randomness:") print(text_with_high_randomness)</pre>	Temperature around 0.5 or a top-p value around 0.7. This is often a good starting point for balanced outputs.

Top K Sampling	<pre># Assuming the library/API is called 'generative_ai' and it supports top-p sampling def generate_text_with_top_p_sampling(prompt, top_p):     """Generates text using the generative AI model with top-p sampling.      Args:         prompt: The text prompt to provide to the model.         top_p: A value between 0 and 1 representing the probability of selecting                the next word from the top p most likely words at each step.                Higher values lead to less randomness.      Returns:         The generated text by the model.     """     try:         generated_text = generative_ai.generate(             prompt=prompt,             top_p=top_p         )         return generated_text     except (AttributeError, NotImplementedError):         print("Top-p sampling not supported by this library/API.")         return None  # Example Usage user_prompt = "Write a new article about a scientific breakthrough." text_with_low_probability = generate_text_with_top_p_sampling(user_prompt, 0.5) text_with_high_probability = generate_text_with_top_p_sampling(user_prompt, 0.9)  print("Text with lower probability sampling:") print(text_with_low_probability)  print("Text with higher probability sampling:") print(text_with_high_probability)</pre>	<p>Creativity: Higher k values (100+) encourage exploration and potentially more surprising outputs.</p> <p>Coherence: Lower k values (1-10) promote focus and potentially more grammatically correct and consistent text.</p>
----------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Pre-trained Generative AI Models from Google AI Like BERT (Bidirectional Encoder Representations from Transformers).	

### Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Pre-trained Generative AI Models from Google AI	The pre-trained Generative AI models accessible through APIs or libraries can help the user to convert various NLP tasks, potentially including text generation when combined with other techniques.

