Model Development Phase Template

Date	15 July 2024
Team ID	SWTID1720259116
Project Title	Nutrition App Using Gemini Pro: Your Comprehensive Guide To Healthy Eating And Well-Being
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be documented and available for review. This documentation will detail the specific steps involved in training the model, including hyperparameter tuning, data pre-processing techniques, and the chosen optimization algorithm. The model validation and evaluation process will be thoroughly presented, outlining the evaluation metrics employed (e.g., classification reports, accuracy, confusion matrices) and the results obtained for different model configurations. This comprehensive report will provide insights into the model's performance and aid in future optimization efforts.

Initial Model Training Code:

```python
from dotenv import load_dotenv
load_dotenv()
import streamlit as st
import os
import google.generativeai as genai
from PIL import Image

genai.configure(api_key = os.getenv("AIzaSyDlOHFLmgqFR6GZfdpCbZoAxpT1vgeszfk"))


##``
def get_gemini_response(input,image,prompt):
    model = genai.GenerativeModel('gemini-pro-vision')
    response = model.generate_content([input,image[0],prompt])
    return response.text


def input_image_setup(uploaded_file):
    if uploaded_file is not None:
        bytes_data = uploaded_file.getvalue()

        image_parts = [
            {
                "mime_type": uploaded_file.type,
                "data": bytes_data
            }
        ]
        return image_parts
    else:
        raise FileNotFoundError("no file uploaded")

```

- **Imports:** The code starts by importing libraries like `streamlit` (for building web apps), `os` (for interacting with the operating system), and `google.generativeai` (likely a custom library for interacting with Gemini generative AI model).
- **`get_gemini_response` function:** This function appears to take three inputs: user input text, an image, and a prompt. It then likely uses the Gemini generative AI model to process the input and generate a response.

- **`Input image setup` function:** This function seems to handle uploaded images. It checks if an

image file was uploaded and if so, extracts the data from the uploaded file. If no image is uploaded, it raises a `FileNotFoundError`.

```
35  ##
36  input_prompt = """you are an expert in nutritionist where you need to see the food items from the image and calculate the total calories, also provide the details of every food in below format:
37
38  1. Item 1 - no of calories
39  2. Item 2 - no of calories
40  ----
41  ----
42  """
43
44
45  st.set_page_config(page_title = "app")
46  st.header("ai app")
47  input = st.text_input("Input Prompt : ",key ="input")
48  uploaded_file = st.file_uploader("choose an image..", type = ["jpg","jpeg","png"])
49  image = ""
50  if uploaded_file is not None:
51      image = Image.open(uploaded_file)
52      st.image(image,caption="uploaded image.",use_column_width = True)
53  submit = st.button("tell me the total calories")
54
55
56  if submit:
57      image_data = input_image_setup(uploaded_file)
58      response = get_gemini_response(input_prompt,image_data,input)
59      st.subheader("the response is :")
60      st.write(response)
61
```

- Lines 37-39 define a variable named `input_prompt` with a string that instructs the user to enter a question for a nutritionist.
- Lines 42-44 set up the page title and header for the application using the `st.set_page_config` and `st.header` functions from a library likely called `streamlit`.
- Lines 44-49 allow the user to input text using the `st.text_input` function and stores it in a variable named `input`.
- Lines 49-53 enable the user to upload an image using the `st.file_uploader` function. It stores the uploaded image data in a variable named `uploaded_file`. The code then displays the uploaded image on the screen.
- Lines 53-59 define a button element named `submit`. If the user clicks the button, the code calls the `get_gemini_response` function (which isn't shown in the screenshot) and displays the response using `st.write`.

In essence, this code snippet creates a simple user interface for a nutrition app that allows users to ask questions and potentially upload images for a more comprehensive analysis.