```python
!pip install pytesseract opencv-python pandas numpy tensorflow

import cv2
import pytesseract
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
Collecting pytesseract
  Downloading pytesseract-0.3.13-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: opencv-python in
/usr/local/lib/python3.11/dist-packages (4.11.0.86)
Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in
/usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: tensorflow in
/usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.11/dist-packages (from pytesseract) (24.2)
Requirement already satisfied: Pillow>=8.0.0 in
/usr/local/lib/python3.11/dist-packages (from pytesseract) (11.1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (5.29.4)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in
```

```
/usr/local/lib/python3.11/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: h5py>=3.11.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.11/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.45.1)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-
packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in
/usr/local/lib/python3.11/dist-packages (from keras>=3.5.0-
>tensorflow) (0.0.8)
Requirement already satisfied: optree in
/usr/local/lib/python3.11/dist-packages (from keras>=3.5.0-
>tensorflow) (0.14.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18-
>tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.11/dist-packages (from
tensorboard<2.19,>=2.18->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18-
>tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.19,>=2.18->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0-
>tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0-
>tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in
/usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0-
>rich->keras>=3.5.0->tensorflow) (0.1.2)
Downloading pytesseract-0.3.13-py3-none-any.whl (14 kB)
Installing collected packages: pytesseract
Successfully installed pytesseract-0.3.13
```

```
!apt-get install -y tesseract-ocr
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  tesseract-ocr-eng tesseract-ocr-osd
The following NEW packages will be installed:
  tesseract-ocr tesseract-ocr-eng tesseract-ocr-osd
0 upgraded, 3 newly installed, 0 to remove and 29 not upgraded.
Need to get 4,816 kB of archives.
After this operation, 15.6 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-
ocr-eng all 1:4.00~git30-7274cfa-1.1 [1,591 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-
ocr-osd all 1:4.00~git30-7274cfa-1.1 [2,990 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-
ocr amd64 4.1.1-2.1build1 [236 kB]
Fetched 4,816 kB in 0s (11.0 MB/s)
Selecting previously unselected package tesseract-ocr-eng.
(Reading database ... 126209 files and directories currently
installed.)
Preparing to unpack .../tesseract-ocr-eng_1%3a4.00~git30-7274cfa-
1.1_all.deb ...
Unpacking tesseract-ocr-eng (1:4.00~git30-7274cfa-1.1) ...
Selecting previously unselected package tesseract-ocr-osd.
Preparing to unpack .../tesseract-ocr-osd_1%3a4.00~git30-7274cfa-
1.1_all.deb ...
Unpacking tesseract-ocr-osd (1:4.00~git30-7274cfa-1.1) ...
Selecting previously unselected package tesseract-ocr.
```

```
Preparing to unpack .../tesseract-ocr_4.1.1-2.1build1_amd64.deb ...
Unpacking tesseract-ocr (4.1.1-2.1build1) ...
Setting up tesseract-ocr-eng (1:4.00~git30-7274cfa-1.1) ...
Setting up tesseract-ocr-osd (1:4.00~git30-7274cfa-1.1) ...
Setting up tesseract-ocr (4.1.1-2.1build1) ...
Processing triggers for man-db (2.10.2-1) ...
```

```python
# Load image
image_path = "/content/download.jpeg"  # Change path if necessary
img = cv2.imread(image_path)

# Convert image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply OCR to extract text
text = pytesseract.image_to_string(gray)

# Print extracted text
print("Extracted Ingredients:\n", text)
```

```
Extracted Ingredients:
 INGREDIENTS: SUGAR, ENRICHED BLEACHED FLOUR (WHEAT
‘FLOUR, NIACIN, REDUCED IRON, THIAMIN MONONITRATE,
RIBOFLAVIN, FOLIC ACID), SEMI-SWEET CHOCOLATE CHIPS
(SUGAR, CHOCOLATE LIQUOR, COCOA BUTTER, SOY LECITHIN
-MULSIFIER), VANILLA), COCOA (PROCESSED WITH ALKAL),
NOLA OR SOYBEAN OIL. BITTERSWEET CHOCOLATE CHIPS
(CHOCOLATE LIQUOR, SUGAR, COCOA BUTTER, MILK FAT,
SOY LECITHIN (EMULSIFIER), VANILLA), MILK CHOCOLATE
CCHIPS (SUGAR, WHOLE MILK POWDER, CHOCOLATE LIQUOR,
COCOA BUTTER, SOY LECITHIN EMULSIFER, VANILLA), SALT,
ARTIFICIAL FLAVOR, SODIUM BICARBONATE.
```

```python
# Sample dataset for matching (Can be expanded)
ingredient_data = {
    "Ingredient": ["Sugar", "Flour", "Chocolate Chips", "Cocoa", 
"Oil", "Salt", "Artificial Flavor"],
    "Calories": [400, 365, 500, 230, 900, 0, 0],
    "Fat": [0, 1, 30, 2, 100, 0, 0],
    "Carbs": [100, 80, 70, 20, 0, 0, 0],
    "Protein": [0, 10, 5, 5, 0, 0, 0],
    "Sodium": [0, 0, 5, 2, 0, 400, 0],
    "Sugar Content": [100, 30, 80, 10, 0, 0, 0],
    "Additives": [0, 0, 10, 0, 0, 0, 50]
}

df = pd.DataFrame(ingredient_data)

# Display DataFrame
df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 7,\n  \"fields\": [\n
{\n      \"column\": \"Ingredient\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"Sugar\",\n          \"Flour\",\n
\"Salt\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Calories\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 312,\n        \"min\": 0,\n
\"max\": 900,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          400,\n          365,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Fat\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 37,\n        \"min\": 0,\n
\"max\": 100,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          1,\n          100,\n          30\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Carbs\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 43,\n
\"min\": 0,\n        \"max\": 100,\n        \"num_unique_values\": 5,\
n        \"samples\": [\n          80,\n          0,\n          70\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Protein\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 10,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          0,\n
10,\n          5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sodium\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 150,\n        \"min\": 0,\n        \"max\": 400,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          5,\n
400,\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sugar Content\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 41,\n        \"min\": 0,\n
\"max\": 100,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          30,\n          0,\n          80\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Additives\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
18,\n        \"min\": 0,\n        \"max\": 50,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          0,\n
10,\n          50\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```python
# Prepare input (X) and output (Y)
X = df.drop(columns=["Ingredient"])
y = np.random.uniform(1, 10, len(X))  # Generate random health scores
(Replace with actual scores if available)
```

```python
# Normalize Data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Build ANN Model
model = Sequential([
    Dense(16, activation='relu', input_shape=(X_scaled.shape[1],)),
    Dense(8, activation='relu'),
    Dense(1, activation='linear')  # Output: Health Score (1-10)
])

# Compile Model
model.compile(optimizer='adam', loss='mse')

# Train Model
model.fit(X_scaled, y, epochs=100, verbose=1)

# Save Model
model.save("food_health_model.h5")
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/
dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Epoch 1/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - loss: 45.4359
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - loss: 45.3008
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 45.1662
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - loss: 45.0372
Epoch 5/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 44.9096
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - loss: 44.7827
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 44.6563
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 44.5251
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - loss: 44.3923
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 44.2617
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - loss: 44.1328
```

```
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step - loss: 44.0060
Epoch 13/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - loss: 43.8794
Epoch 14/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - loss: 43.7553
Epoch 15/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 43.6319
Epoch 16/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step - loss: 43.5087
Epoch 17/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 43.3856
Epoch 18/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 43.2626
Epoch 19/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 43.1398
Epoch 20/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 43.0170
Epoch 21/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 42.8954
Epoch 22/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - loss: 42.7800
Epoch 23/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 42.6645
Epoch 24/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 42.5487
Epoch 25/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - loss: 42.4328
Epoch 26/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - loss: 42.3237
Epoch 27/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - loss: 42.2239
Epoch 28/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 42.1246
Epoch 29/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 42.0237
Epoch 30/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 41.9222
Epoch 31/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 41.8203
Epoch 32/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 41.7179
Epoch 33/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - loss: 41.6151
Epoch 34/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 41.5118
Epoch 35/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 41.4082
Epoch 36/100
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 41.3077
Epoch 37/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 41.2064
Epoch 38/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 41.1043
Epoch 39/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 41.0012
Epoch 40/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 40.8959
Epoch 41/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - loss: 40.7901
Epoch 42/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 40.6838
Epoch 43/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 40.5767
Epoch 44/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 40.4689
Epoch 45/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - loss: 40.3603
Epoch 46/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - loss: 40.2510
Epoch 47/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 40.1410
Epoch 48/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 40.0302
Epoch 49/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 39.9183
Epoch 50/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 39.7945
Epoch 51/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 39.6683
Epoch 52/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 39.5399
Epoch 53/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 39.4095
Epoch 54/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 39.2771
Epoch 55/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 39.1430
Epoch 56/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 39.0072
Epoch 57/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 38.8665
Epoch 58/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - loss: 38.7197
Epoch 59/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - loss: 38.5697
Epoch 60/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 38.4162
```

```
Epoch 61/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step - loss: 38.2563
Epoch 62/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - loss: 38.0934
Epoch 63/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - loss: 37.9281
Epoch 64/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - loss: 37.7606
Epoch 65/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - loss: 37.5911
Epoch 66/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - loss: 37.4204
Epoch 67/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - loss: 37.2483
Epoch 68/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - loss: 37.0742
Epoch 69/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - loss: 36.8984
Epoch 70/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - loss: 36.7217
Epoch 71/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - loss: 36.5424
Epoch 72/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - loss: 36.3615
Epoch 73/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - loss: 36.1784
Epoch 74/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 147ms/step - loss: 35.9932
Epoch 75/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - loss: 35.8063
Epoch 76/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - loss: 35.6198
Epoch 77/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step - loss: 35.4336
Epoch 78/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - loss: 35.2458
Epoch 79/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - loss: 35.0563
Epoch 80/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step - loss: 34.8653
Epoch 81/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 34.6727
Epoch 82/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - loss: 34.4782
Epoch 83/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 34.2820
Epoch 84/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 34.0835
Epoch 85/100
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 33.8824
Epoch 86/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 33.6794
Epoch 87/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 33.4746
Epoch 88/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - loss: 33.2680
Epoch 89/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 33.0597
Epoch 90/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - loss: 32.8496
Epoch 91/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 32.6365
Epoch 92/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - loss: 32.4204
Epoch 93/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - loss: 32.1982
Epoch 94/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 31.9738
Epoch 95/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 31.7472
Epoch 96/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 31.5181
Epoch 97/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 31.2859
Epoch 98/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - loss: 31.0514
Epoch 99/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 30.8148
Epoch 100/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 30.5761

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

# Function to Predict Scores
def predict_health_score(ingredient):
    if ingredient in df["Ingredient"].values:
        values = df[df["Ingredient"] ==
ingredient].drop(columns=["Ingredient"]).values
        values_scaled = scaler.transform(values)
        return model.predict(values_scaled)[0][0]  # Return predicted
score
    else:
        return 5  # Default score for unknown ingredients

# Apply Prediction to Extracted Ingredients
```

```python
extracted_ingredients = ["Sugar", "Flour", "Chocolate Chips", "Cocoa",
"Oil", "Salt"]  # Replace with actual extracted ingredients

ingredient_scores = {ingredient: predict_health_score(ingredient) for
ingredient in extracted_ingredients}
print("Ingredient Health Scores:", ingredient_scores)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but MinMaxScaler was fitted with feature names
  warnings.warn(

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 349ms/step

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but MinMaxScaler was fitted with feature names
  warnings.warn(

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but MinMaxScaler was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:27
39: UserWarning: X does not have valid feature names, but MinMaxScaler
was fitted with feature names
  warnings.warn(

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but MinMaxScaler was fitted with feature names
  warnings.warn(

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
Ingredient Health Scores: {'Sugar': np.float32(1.0773727), 'Flour':
np.float32(1.2051194), 'Chocolate Chips': np.float32(1.0413492),
'Cocoa': np.float32(0.7929162), 'Oil': np.float32(1.3794785), 'Salt':
np.float32(1.9315897)}

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but MinMaxScaler was fitted with feature names
  warnings.warn(
```

```python
# Calculate Final Score (Average of Ingredient Scores)
final_score = np.mean(list(ingredient_scores.values()))

# Classify Food Health
if final_score >= 7:
    classification = "Good"
elif 4 <= final_score < 7:
    classification = "Moderate"
else:
    classification = "Bad"

print(f"Final Food Health Score: {final_score:.2f}/10 -
Classification: {classification}")
```

```
Final Food Health Score: 1.24/10 - Classification: Bad
```