


```
# Step 1: Install required libraries
!sudo apt install tesseract-ocr
!pip install pytesseract pillow opencv-python scikit-learn pandas numpy
```

 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 1s (3,305 kB/s)
 debconf: unable to initialize frontend: Dialog
 debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/
 debconf: falling back to frontend: Readline
 debconf: unable to initialize frontend: Readline
 debconf: (This frontend requires a controlling tty.)
 debconf: falling back to frontend: Teletype
 dpkg-preconfigure: unable to re-open stdin:
 Selecting previously unselected package tesseract-ocr-eng.
 (Reading database ... 125044 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) ...
 Collecting pytesseract
 Downloading pytesseract-0.3.13-py3-none-any.whl.metadata (11 kB)
 Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (11.1.0)
 Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-packages (4.11.0.86)
 Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
 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 (1.26.4)
 Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from pytesseract) (24.2)
 Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
 Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
 Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.5.0)
 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: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
 Downloading pytesseract-0.3.13-py3-none-any.whl (14 kB)
 Installing collected packages: pytesseract
 Successfully installed pytesseract-0.3.13

```
import pytesseract
from PIL import Image
import cv2
import re
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from google.colab import files
```

```
uploaded = files.upload()
```

 No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to

```
def extract_text_from_image(image_path):
    # Load the image
```

```

image = Image.open(image_path)

# Convert to grayscale (optional, improves OCR accuracy)
image = image.convert("L")

# Use pytesseract to extract text
text = pytesseract.image_to_string(image)

return text

def clean_ocr_text(text):
    # Remove special characters and extra spaces
    text = re.sub(r"^[a-zA-Z0-9\s]", "", text)
    text = re.sub(r"\s+", " ", text).strip()

    # Convert to lowercase
    text = text.lower()

    return text

ocr_texts = []
for file_name in uploaded.keys():
    ocr_text = extract_text_from_image(file_name)
    cleaned_text = clean_ocr_text(ocr_text)
    ocr_texts.append(cleaned_text)
    print(f"File: {file_name}\nExtracted Text: {cleaned_text}\n")

File: labeled_image.png
Extracted Text: ingredients enriched unbleached flour wheat flour malted barley flour ascorbic acid dough conditioner niacin reduced irc

vectorizer = TfidfVectorizer(max_features=50) # Adjust max_features as needed
ocr_features = vectorizer.fit_transform(ocr_texts).toarray()

import pandas as pd

# Load the dataset (replace with your actual dataset)
df = pd.read_csv("/content/large_allergen_food_dataset (1).csv")

# Extract features and labels
# Features: Ingredients, Detected Allergens, Risk Level (encoded as numerical values)
# Labels: Risk Level (encoded as 0 for Medium Risk, 1 for High Risk)
from sklearn.preprocessing import LabelEncoder

# Encode Risk Level as numerical labels
label_encoder = LabelEncoder()
df["Risk Level Encoded"] = label_encoder.fit_transform(df["Risk Level"]) # 0: Medium Risk, 1: High Risk

existing_features = df[["Ingredients", "Detected Allergens"]]

# Convert text features (Ingredients, Detected Allergens) into numerical features using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

# Combine Ingredients and Detected Allergens into a single text feature
df["Combined Text"] = df["Ingredients"] + " " + df["Detected Allergens"]

# Fit TF-IDF vectorizer on the combined text
vectorizer_existing = TfidfVectorizer(max_features=50) # Adjust max_features as needed
existing_features_tfidf = vectorizer_existing.fit_transform(df["Combined Text"]).toarray()

# Labels
labels = df["Risk Level Encoded"]

# Reshape ocr_features to have the same number of rows as existing_features_tfidf
ocr_features = ocr_features[0] # Extract the first (and only) row from ocr_features
# Assuming existing_features_tfidf is (10000,50), and ocr_features is (1,43), so we want to repeat the rows of ocr_features.
# to get it to (10000,43)
# If they have different columns, it will raise a different error.
num_rows = existing_features_tfidf.shape[0] # Get the number of rows from existing_features_tfidf
ocr_features = np.tile(ocr_features, (num_rows, 1)) # Repeat the values of ocr_features into 10000 rows

```

```
combined_features = np.hstack((existing_features_tfidf, ocr_features))





combined_features = np.hstack((existing_features_tfidf, ocr_features))

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(combined_features, labels, test_size=0.2, random_state=42)

# Step 11: Train the Random Forest model
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

  RandomForestClassifier  

```
RandomForestClassifier(random_state=42)
```

```
from sklearn.metrics import accuracy_score

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
```

 Model Accuracy: 1.0

```
# Define allergens_map
allergens_map = {
    "milk": "Almond Milk, Cashew Cream, Soy Milk",
    "wheat": "Gluten-Free Flour, Rice Flour",
    "soy": "Coconut Aminos, Tamari Sauce",
    "peanuts": "Sunflower Butter, Almond Butter",
    "shellfish": "Mushrooms, Tofu",
    "eggs": "Chia Seeds, Flaxseeds",
    "gluten": "Gluten-Free Wrap, Rice Paper",
    "tree nuts": "Seeds (Sunflower, Pumpkin)"
}

def predict_from_image(image_path):
    # Extract and clean OCR text
    ocr_text = extract_text_from_image(image_path)
    cleaned_text = clean_ocr_text(ocr_text)

    # Convert OCR text to features
    ocr_features = vectorizer.transform([cleaned_text]).toarray()

    # Combine with existing features
    new_existing_features = vectorizer_existing.transform([cleaned_text]).toarray()
    new_combined_features = np.hstack((new_existing_features, ocr_features))


    # Make prediction
    prediction = model.predict(new_combined_features)
    risk_level = label_encoder.inverse_transform(prediction)[0] # Convert back to original label

    # Generate notes and alternative suggestions
    detected_allergens = ", ".join([word for word in cleaned_text.split() if word in allergens_map])
    warning = f"⚠️ {risk_level}: Contains {detected_allergens}. Avoid if allergic!"
    alternatives = ", ".join([allergens_map[allergen] for allergen in detected_allergens.split(", ") if allergen in allergens_map])

    return risk_level, warning, alternatives

# Upload new images
new_uploaded = files.upload()

# Process each uploaded image
for new_file_name in new_uploaded.keys():
    risk_level, warning, alternatives = predict_from_image(new_file_name)
    print(f"File: {new_file_name}")
    print(f"Risk Level: {risk_level}")
    print(f"Notes: {warning}")
    print(f"Alternative Suggestions: {alternatives}\n")
```

 No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving test_image.jfif to test_image (3).jfif
File: test_image (3).jfif
Risk Level: Medium Risk
Notes: ⚠ Medium Risk: Contains wheat, soy, milk, soy. Avoid if allergic!
Alternative Suggestions: Gluten-Free Flour. Rice Flour. Coconut Aminos. Tamari Sauce. Almond Milk. Cashew Cream. Soy Milk. Coconut Amino