VNR Vignana Jyothi Institute of Engineering and Technology (Affiliated to J.N.T.U, Hyderabad) Bachupally(v), Hyderabad, Telangana, India.

A course project submitted in complete requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND BUSINESS SYSTEMS

Submitted by

G. Tejaswi	**21071A3219**
M.V. Swathi	**21071A3244**
M. Sravanth	**21071A3245**
M.A. Sami	**21071A3257**
P. Deepika	**21071A3255**

Under the guidance of

Mrs. P. Jyothi

Assistant Professor

Department of Computer Science and Engineering

VNR Vignana Jyothi Institute of Engineering and Technology
(Affiliated to J.N.T.U, Hyderabad) Bachupally(v), Hyderabad, Telangana,India.

CERTIFICATE

This is to certify that

G.Tejaswi (21071A3219),
M.V.Swathi (21071A3244),
M. Sravanth (21071A3245),
M.A. Sami (21071A3247),
P. Deepika (21071A3255)

have completed their course project work at CSE Department of VNR VJIET, Hyderabad entitled **"SkinCare AI: personalized cosmetic suitability analyser".** This work is carried out under my supervision and has not beensubmitted to any other University/Institute for award of any degree/diploma.

Mrs. P. Jyothi
Assistant Professor
CSE Department
VNRVJIET

Dr. S. Nagini
Head Of Department
CSE & CSBS Department
VNRVJIET

DECLARATION

This is to certify that our project report titled “Machine Learning Algorithms for Intrusion detection system” submitted to Vallurupalli Nageswara Rao Institute of Engineering and Technology in complete fulfilment of requirement for the award of Bachelor of Technology in Computer Science and Engineering is a Bonafide report to the work carried outby us under the guidance and supervision of Mrs. P. Jyothi, Assistant Professor, Department of Computer Science and Engineering, Vallurupalli Nageswara Rao Institute of Engineering and Technology. To the best of our knowledge, this has not been submitted in any form to other university or institution for the award of any degree or diploma.

ABSTRACT

The "SkinCare AI: personalized cosmetic suitability analyser" project is a personalized cosmetic suitability analyzer that leverages machine learning techniques to predict the suitability of cosmetic products for various skin types. The system accepts input in the form of an image containing the list of ingredients of a cosmetic product and the user's skin type. Using image processing and optical character recognition (OCR) techniques, the system extracts the ingredients from the image. These ingredients, along with the user's skin type, are then fed into pre-trained machine learning models. The models are trained on a dataset of cosmetic products and their suitability scores for different skin types. Based on this input, the system predicts the suitability score of the given product for the specified skin type. The user receives the suitability score as the output, providing valuable insights into the compatibility of the cosmetic product with their skin type.

INTRODUCTION

The skincare industry is inundated with a plethora of cosmetic products, each claiming to offer solutions for various skin concerns. However, amidst the abundance of options, consumers often find themselves overwhelmed and confused when it comes to selecting products that are compatible with their specific skin type. This dilemma is exacerbated by the lack of personalized guidance and the complexity of deciphering product labels, particularly when it comes to understanding the ingredients used.

As a result, consumers are left to navigate through a maze of skincare products, often resorting to trial and error to find what works best for their skin. This approach not only leads to frustration but can also exacerbate existing skin issues and cause adverse reactions.

In light of these challenges, there is a growing need for innovative solutions that empower consumers with personalized skincare recommendations tailored to their unique skin type and concerns. By leveraging advancements in artificial intelligence and image processing technologies, such solutions have the potential to revolutionize the way individuals approach skincare, providing them with the guidance and confidence they need to make informed decisions about the products they use on their skin.

Proposed System:

The proposed system, named "SkinCare AI," aims to revolutionize the way individuals approach skincare by offering personalized recommendations for cosmetic products based on their unique skin type. The system leverages advanced technologies such as image processing and machine learning to analyze cosmetic product ingredients and predict their compatibility with different skin types.

Below are the key components and functionalities of the proposed system:

1. Image Processing Module:

- This module is responsible for processing images containing cosmetic product labels.
- It employs techniques such as optical character recognition (OCR) to extract the list of ingredients from the images.
- Advanced image processing algorithms are utilized to enhance image quality and improve ingredient extraction accuracy.

2. Machine Learning Module:

- The machine learning module comprises pre-trained models that have been trained on a dataset of cosmetic products and their suitability scores for various skin types.
- These models are capable of predicting the suitability score of a given cosmetic product for a specified skin type based on its ingredients.
- Different machine learning models may be employed to predict suitability scores for different skin types (e.g., combination, dry, oily, sensitive).

3. User Interface:

- The system features a user-friendly interface that allows users to input images of cosmetic product labels and specify their skin type.
- Users are provided with intuitive controls to interact with the system and receive personalized recommendations.

- The interface may be accessible via web-based applications, mobile apps, or desktop software, depending on user preferences and requirements.

4. Recommendation Engine:

- Based on the extracted ingredients and the user's specified skin type, the recommendation engine calculates the suitability score of the cosmetic product.
- The recommendation engine employs machine learning models to predict suitability scores and ranks cosmetic products based on their compatibility with the user's skin type.
- Users receive personalized recommendations tailored to their unique skincare needs, facilitating informed decision-making when selecting skincare products.

5. Integration and Scalability:

- The proposed system is designed to be scalable and adaptable to accommodate future enhancements and expansions.
- It may be integrated with existing e-commerce platforms, skincare apps, or beauty websites to provide seamless access to personalized skincare recommendations.
- The system architecture allows for easy integration with third-party APIs and services to enhance functionality and user experience.

Overall, the proposed "SkinCare AI" system offers a comprehensive solution to address the challenges faced by consumers in selecting skincare products. By combining the power of image processing, machine learning, and personalized recommendations, the system empowers users to make informed decisions and achieve healthier, happier skin.

Challenges and Future Scope

Ensuring data quality and availability is a significant challenge, requiring access to comprehensive, up-to-date ingredient databases and dealing with variations in label formats and print quality that can affect OCR accuracy. Image processing accuracy is hindered by OCR's limitations with poorly printed labels, cursive fonts, and low-resolution images, compounded by varying lighting and camera quality. Training machine learning models necessitates a high-quality labeled dataset, requiring extensive effort and expertise, with added complexity in ensuring models generalize across diverse skin types and conditions. Personalization is complicated by the wide variability in individual skin reactions to ingredients and the dynamic nature of skin conditions influenced by external factors. Protecting user privacy and data security is crucial, demanding robust measures to handle sensitive information and comply with regulations like GDPR and CCPA. Building user trust involves transparent communication about the recommendation process, while ensuring the interface is intuitive and accessible across various devices is critical for user adoption.

Future enhancements include advanced ingredient analysis, such as evaluating chemical interactions and detecting allergens to provide safer recommendations. Leveraging deep learning techniques can improve prediction accuracy, with continuous learning mechanisms updating models based on user feedback and new dermatological research. Integration with wearable technology offers real-time, context-aware recommendations by monitoring skin conditions and incorporating health data for holistic skincare advice. Expanding globally involves supporting multiple languages and providing region-specific recommendations to address diverse cosmetic products and skin concerns. Collaborating with dermatologists ensures medical accuracy and can integrate teledermatology services for professional consultations alongside AI-driven advice. Developing community features allows users to share experiences and feedback, continuously enhancing the recommendation system's accuracy and relevance. Partnering with e-commerce platforms enables direct product purchases and personalized marketing, offering targeted promotions to enhance user experience and engagement.

Code Snippet

In [2]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import joblib
```

In [3]:
```
df = pd.read_csv('C:/Users/TLS DEEPIKA/Desktop/cosmetics.csv')
df
```

Out[3]:

	Label	Brand	Name	Price	Rank	Ingredients	Combinat
0	Moisturizer	LA MER	Crème de la Mer	175	4.1	Algae (Seaweed) Extract, Mineral Oil, Petrolat...	
1	Moisturizer	SK-II	Facial Treatment Essence	179	4.1	Galactomyces Ferment Filtrate (Pitera), Butyle...	
2	Moisturizer	DRUNK ELEPHANT	Protini™ Polypeptide Cream	68	4.4	Water, Dicaprylyl Carbonate, Glycerin, Ceteary...	
3	Moisturizer	LA MER	The Moisturizing Soft Cream	175	3.8	Algae (Seaweed) Extract, Cyclopentasiloxane, P...	
4	Moisturizer	IT COSMETICS	Your Skin But Better™ CC+™ Cream with SPF 50+	38	4.1	Water, Snail Secretion Filtrate, Phenyl Trimet...	
...	...	...	...	...	...	...	
1467	Sun protect	KORRES	Yoghurt Nourishing Fluid Veil Face Sunscreen B...	35	3.9	Water, Alcohol Denat., Potassium Cetyl Phospha...	
1468	Sun protect	KATE SOMERVILLE	Daily Deflector™ Waterlight Broad Spectrum SPF...	48	3.6	Water, Isododecane, Dimethicone, Butyloctyl Sa...	
1469	Sun protect	VITA LIBERATA	Self Tan Dry Oil SPF 50	54	3.5	Water, Dihydroxyacetone, Glycerin, Sclerocarya...	
1470	Sun protect	ST. TROPEZ TANNING ESSENTIALS	Pro Light Self Tan Bronzing Mist	20	1.0	Water, Dihydroxyacetone, Propylene Glycol, PPG...	
1471	Sun protect	DERMAFLASH	DERMAPROTECT Daily Defense Broad Spectrum SPF 50+	45	0.0	Visit the DERMAFLASH boutique	

1472 rows × 11 columns

In [4]:
```
tfidf = TfidfVectorizer(stop_words='english')
X = tfidf.fit_transform(df['Ingredients'])
```

In [5]:
```
y_combination = df['Combination']
y_dry = df['Dry']
y_normal = df['Normal']
y_oily = df['Oily']
y_sensitive = df['Sensitive']
```

In [6]:
```
X_train_comb, X_test_comb, y_train_comb, y_test_comb = train_test_split(X,
X_train_dry, X_test_dry, y_train_dry, y_test_dry = train_test_split(X, y_dr
X_train_norm, X_test_norm, y_train_norm, y_test_norm = train_test_split(X,
X_train_oily, X_test_oily, y_train_oily, y_test_oily = train_test_split(X,
X_train_sens, X_test_sens, y_train_sens, y_test_sens = train_test_split(X,
```

In [7]:
```
model_comb = RandomForestRegressor(n_estimators=100, random_state=42)
model_comb.fit(X_train_comb, y_train_comb)
```

Out[7]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [8]:
```
model_dry = RandomForestRegressor(n_estimators=100, random_state=42)
model_dry.fit(X_train_dry, y_train_dry)
```

Out[8]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [9]:
```
model_norm = RandomForestRegressor(n_estimators=100, random_state=42)
model_norm.fit(X_train_norm, y_train_norm)
```

Out[9]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [10]:
```
model_oily = RandomForestRegressor(n_estimators=100, random_state=42)
model_oily.fit(X_train_oily, y_train_oily)
```

Out[10]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [11]:
```
model_sens = RandomForestRegressor(n_estimators=100, random_state=42)
model_sens.fit(X_train_sens, y_train_sens)
```

Out[11]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [12]:
```
y_pred_comb = model_comb.predict(X_test_comb)
print(f'Combination - Mean Squared Error: {mean_squared_error(y_test_comb,

y_pred_dry = model_dry.predict(X_test_dry)
print(f'Dry - Mean Squared Error: {mean_squared_error(y_test_dry, y_pred_dr

y_pred_norm = model_norm.predict(X_test_norm)
print(f'Normal - Mean Squared Error: {mean_squared_error(y_test_norm, y_pre

y_pred_oily = model_oily.predict(X_test_oily)
print(f'Oily - Mean Squared Error: {mean_squared_error(y_test_oily, y_pred_

y_pred_sens = model_sens.predict(X_test_sens)
print(f'Sensitive - Mean Squared Error: {mean_squared_error(y_test_sens, y_
```

```
Combination - Mean Squared Error: 0.19154198127452257
Dry - Mean Squared Error: 0.20101641822273428
Normal - Mean Squared Error: 0.18822685691038443
Oily - Mean Squared Error: 0.2050814818423538
Sensitive - Mean Squared Error: 0.2132174929409475
```

In [13]:
```
joblib.dump(model_comb, 'model_comb.pkl')
joblib.dump(model_dry, 'model_dry.pkl')
joblib.dump(model_norm, 'model_norm.pkl')
joblib.dump(model_oily, 'model_oily.pkl')
joblib.dump(model_sens, 'model_sens.pkl')
joblib.dump(tfidf, 'tfidf_vectorizer.pkl')
```

Out[13]:
```
['tfidf_vectorizer.pkl']
```

In [19]:
```
pip install pillow pytesseract
```

```
Requirement already satisfied: pillow in c:\users\tls deepika\anaconda3\li
b\site-packages (9.4.0)
Requirement already satisfied: pytesseract in c:\users\tls deepika\anacond
a3\lib\site-packages (0.3.10)
Requirement already satisfied: packaging>=21.3 in c:\users\tls deepika\ana
conda3\lib\site-packages (from pytesseract) (23.1)
Note: you may need to restart the kernel to use updated packages.
```

In [20]:
```
from PIL import Image, ImageEnhance, ImageFilter
import pytesseract
```

In [21]:
```
def preprocess_image(image_path):
    with Image.open(image_path) as img:
        img = img.convert('L')
        img = img.filter(ImageFilter.SHARPEN)
        enhancer = ImageEnhance.Contrast(img)
        img = enhancer.enhance(2)
        return img
```

In [22]:
```
def read_ingredients(image_path):
    img = preprocess_image(image_path)
    text = pytesseract.image_to_string(img)
    return text
```

In [24]:
```
import os
from PIL import Image, ImageEnhance, ImageFilter
import pytesseract

# Ensure Tesseract executable is in your PATH or specify the location direc
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\te

def preprocess_image(image_path):
    try:
        with Image.open(image_path) as img:
            img = img.convert('L')  # Convert to grayscale
            img = img.filter(ImageFilter.SHARPEN)  # Sharpen the image
            enhancer = ImageEnhance.Contrast(img)
            img = enhancer.enhance(2)  # Enhance contrast
            return img
    except Exception as e:
        print(f"Error in preprocessing image: {e}")
        return None

def read_ingredients(image_path):
    try:
        img = preprocess_image(image_path)
        if img:
            text = pytesseract.image_to_string(img)
            return text
        else:
            return ""
    except Exception as e:
        print(f"Error in reading ingredients from image: {e}")
        return ""
```

```
# Example usage
if __name__ == "__main__":
    # Specify the path to your image file here
    image_path = 'C:/Users/TLS DEEPIKA/Desktop/nivea.jpg'  # Example: 'C:/p
    ingredients_text = read_ingredients(image_path)
    if ingredients_text:
        print(f'Extracted Ingredients: {ingredients_text}')
    else:
        print('Failed to extract ingredients from the image.')
```

```
Extracted Ingredients: INGREDIENTS

Water/Eau, Mineral Oit/Huile minérale,
Microcrystalline Wax/Cire
microcrystalline, Glycerin, Lanolin
Alcohol (Eucerit), Paraffin, Panthenol,
Magnesium Sulfate, Decyl Oleate,
Octyldodecanol, Aluminum Stearates,
Citric Acid, Magnesium Stearate,
Fragrance/Parfum.
```

In [27]:
```
import joblib

def predict_suitability(image_path, user_skin_type):
    # Load the models and TF-IDF vectorizer
    model_comb = joblib.load('model_comb.pkl')
    model_dry = joblib.load('model_dry.pkl')
    model_norm = joblib.load('model_norm.pkl')
    model_oily = joblib.load('model_oily.pkl')
    model_sens = joblib.load('model_sens.pkl')
    tfidf = joblib.load('tfidf_vectorizer.pkl')

    # Read ingredients from image
    ingredients_text = read_ingredients(image_path)

    # Prepare the input vector
    input_vector = tfidf.transform([ingredients_text])

    # Predict suitability scores for each skin type
    scores = {
        "Combination": model_comb.predict(input_vector)[0],
        "Dry": model_dry.predict(input_vector)[0],
        "Normal": model_norm.predict(input_vector)[0],
        "Oily": model_oily.predict(input_vector)[0],
        "Sensitive": model_sens.predict(input_vector)[0]
    }

    return scores[user_skin_type]

# Example usage
if __name__ == "__main__":
    image_path = 'C:/Users/TLS DEEPIKA/Desktop/cerave_d.jpg'
    user_skin_type = input("Enter your skin type: ")
    suitability_score = predict_suitability(image_path, user_skin_type)
    print(f'The suitability score for the product is: {suitability_score}')
```

```
Enter your skin type: Oily
The suitability score for the product is: 0.39666666666666667
```

In [ ]:

```
  # Get the predicted skin type with the highest score
  best_skin_type = max(scores, key=scores.get)
  suitability_score = scores[user_skin_type]

  print("Suitability score is",suitability_score)
  # Craft a descriptive message based on score
  message = ""
  if suitability_score > 0.8:
    message = f"Great news! This product seems like a **perfect match** for your {user_skin_type} skin.  It's like finding a hid
  elif suitability_score > 0.6:
    message = f"This product might be a **good choice** for your {user_skin_type} skin.  Give it a try and see how it works!"
  elif suitability_score > 0.4:
    message = f"The suitability score for this product is **neutral** for your {user_skin_type} skin. It might still work, so cc
  else:
    message = f"This product might not be the best fit for your {user_skin_type} skin with a score of {suitability_score:.2f}.

  return message

# Create widgets
# Create widgets
skin_type_dropdown = widgets.Dropdown(
  options=["Combination", "Dry", "Normal", "Oily", "Sensitive"],
  description="Select Skin Type:"
)

upload_button = FileUpload(
  accept=".jpg,.jpeg,.png",
  description="Upload Image",
  multiple=False
```

In [29]:
```
import joblib
import ipywidgets as widgets
from IPython.display import display
from ipywidgets import FileUpload
import tkinter as tk

def predict_suitability(image_path, user_skin_type):
  # Load the models and TF-IDF vectorizer (assuming these exist)
  model_comb = joblib.load('model_comb.pkl')
  model_dry = joblib.load('model_dry.pkl')
  model_norm = joblib.load('model_norm.pkl')
  model_oily = joblib.load('model_oily.pkl')
  model_sens = joblib.load('model_sens.pkl')
  tfidf = joblib.load('tfidf_vectorizer.pkl')

  # Read ingredients from image (replace with your implementation)
  ingredients_text = "placeholder_ingredients_text"  # Replace with function to read ingredients

  # Prepare the input vector
  input_vector = tfidf.transform([ingredients_text])

  # Predict suitability scores for each skin type
  scores = {
      "Combination": model_comb.predict(input_vector)[0],
      "Dry": model_dry.predict(input_vector)[0],
      "Normal": model_norm.predict(input_vector)[0],
      "Oily": model_oily.predict(input_vector)[0],
      "Sensitive": model_sens.predict(input_vector)[0]
```

```
  return message

# Create widgets
# Create widgets
skin_type_dropdown = widgets.Dropdown(
  options=["Combination", "Dry", "Normal", "Oily", "Sensitive"],
  description="Select Skin Type:"
)

upload_button = FileUpload(
  accept=".jpg,.jpeg,.png",
  description="Upload Image",
  multiple=False
)

output_label = widgets.Label()

def handle_upload(change):
  if upload_button.value:
    #image_path = list(upload_button.value.keys())[0]
    user_skin_type = skin_type_dropdown.value
    output_label.value = predict_suitability(image_path, user_skin_type)

upload_button.observe(handle_upload, names='value')

# Display widgets
display(skin_type_dropdown, upload_button, output_label)
```

Select Skin... Combination

Upload Image (1)

The suitability score for this product is **neutral** for your Combination skin. It might still work, so consider reading reviews from others with similar skin.

```
Suitability score is 0.5014532967032966
```

CONCLUSION

In conclusion, "SkinCare AI" promises to revolutionize skincare by combining image processing, machine learning, and personalized recommendations. While challenges such as data quality, image processing accuracy, and user privacy must be addressed, the system's potential is immense. Future developments can enhance ingredient analysis, leverage advanced machine learning. By expanding globally and collaborating with dermatologists, "SkinCare AI" can provide medically accurate, context-aware skincare advice. Embracing user feedback and integrating with e-commerce will further refine the system, making informed skincare choices accessible and effective for users worldwide.