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 [ ]: