Title: ***Personalized Medical Recommendation System with Machine Learning***

Description:

Welcome to our cutting-edge Personalized Medical Recommendation System, a powerful platform designed to assist users in understanding and managing their health. Leveraging the capabilities of machine learning, our system analyzes user-input symptoms to predict potential diseases accurately.

*_Importing required modules_*

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
import warnings
warnings.filterwarnings("ignore")
```

*_Loading Dataset_*

In [3]:
```
df = pd.read_csv("Training.csv")
```

*_Viewing samples to check the data_*

In [4]:
```
df.head(5)
```

Out[4]:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	blackheads	s
0	1	1	1	0	0	0	0	0	0	0	...	0	
1	0	1	1	0	0	0	0	0	0	0	...	0	
2	1	0	1	0	0	0	0	0	0	0	...	0	
3	1	1	0	0	0	0	0	0	0	0	...	0	
4	1	1	1	0	0	0	0	0	0	0	...	0	

5 rows × 133 columns

*_In above dataframe, we can see the data is arranged label wise. So we need to shuffle it first_*

In [5]:
```
df = shuffle(df)
```

In [6]:
```
df.head()
```

Out[6]:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	blackheads
1595	0	0	0	0	0	0	0	0	0	0	...	0
4463	0	0	0	0	0	0	0	0	0	0	...	0
1156	0	0	0	0	0	0	0	0	0	0	...	0
1331	0	0	0	0	0	0	0	0	0	0	...	0
167	1	1	0	0	0	0	0	0	0	0	...	0

5 rows × 133 columns

Checking unique values of target variable (Diseases which to be predict)

In [7]:
```
df.prognosis.nunique()
```

Out[7]:
```
41
```

In [8]:
```
X = df.drop("prognosis", axis=1)
y = df.prognosis
```

In [9]:
```
le = LabelEncoder()
y = le.fit_transform(y)
```

Checking the number provided to each prognosis by label encoder

In [10]:
```
print("Label mapping:")
for label, integer_value in zip(le.classes_, le.transform(le.classes_)):
    print(f"{label}: {integer_value}")
```

```
Label mapping:
(vertigo) Paroymsal  Positional Vertigo: 0
AIDS: 1
Acne: 2
Alcoholic hepatitis: 3
Allergy: 4
Arthritis: 5
Bronchial Asthma: 6
Cervical spondylosis: 7
Chicken pox: 8
Chronic cholestasis: 9
Common Cold: 10
Dengue: 11
Diabetes : 12
Dimorphic hemmorhoids(piles): 13
Drug Reaction: 14
Fungal infection: 15
GERD: 16
Gastroenteritis: 17
Heart attack: 18
Hepatitis B: 19
Hepatitis C: 20
Hepatitis D: 21
Hepatitis E: 22
Hypertension : 23
Hyperthyroidism: 24
Hypoglycemia: 25
Hypothyroidism: 26
Impetigo: 27
Jaundice: 28
Malaria: 29
Migraine: 30
Osteoarthristis: 31
Paralysis (brain hemorrhage): 32
Peptic ulcer diseae: 33
Pneumonia: 34
Psoriasis: 35
Tuberculosis: 36
Typhoid: 37
Urinary tract infection: 38
Varicose veins: 39
hepatitis A: 40
```

*_Split the data into 4 Variables, i.e. X_train, X_test, y_train, y_test_*

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=30)
```

In [12]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[12]:
```
((3444, 132), (1476, 132), (3444,), (1476,))
```

*_Tried applying different classifiers to get the best out of it_*

In [13]:
```
models = {
    "SVC": SVC(kernel="linear"),
    "GradientBoost": GradientBoostingClassifier(n_estimators=120, random_state=42),
    "RandomForest": RandomForestClassifier(n_estimators=120, random_state=42),
    "Multinomial" : MultinomialNB(),
    "KNearestNeighbors": KNeighborsClassifier(n_neighbors=5)
}
```

In [14]:
```
for model_name,model in models.items():
    print(model_name, ":", model)
```

```
SVC : SVC(kernel='linear')
GradientBoost : GradientBoostingClassifier(n_estimators=120, random_state=42)
RandomForest : RandomForestClassifier(n_estimators=120, random_state=42)
Multinomial : MultinomialNB()
KNearestNeighbors : KNeighborsClassifier()
```

*_Checking Accuracy, F1_Score, and Confusion matrix for each model_*

In [15]:
```
for model_name, model in models.items():
    model.fit(X_train, y_train)
    pred = model.predict(X_test)

    Accuracy = accuracy_score(y_test, pred)
    F1_score = f1_score(y_test, pred, average="weighted")
    cm = confusion_matrix(y_test, pred)

    print(f"{model_name} Accuarcy {Accuracy}")
    print(f"{model_name} F1 Score{F1_score}")
    print(f"{model_name} Confusion Matrix:")
    print(np.array2string(cm, separator=","))
    print()
```

```
SVC Accuarcy 1.0
SVC F1 Score1.0
SVC Confusion Matrix:
[[34, 0, 0,..., 0, 0, 0],
 [ 0,41, 0,..., 0, 0, 0],
 [ 0, 0,34,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,29, 0, 0],
 [ 0, 0, 0,..., 0,43, 0],
 [ 0, 0, 0,..., 0, 0,34]]

GradientBoost Accuarcy 1.0
GradientBoost F1 Score1.0
GradientBoost Confusion Matrix:
[[34, 0, 0,..., 0, 0, 0],
 [ 0,41, 0,..., 0, 0, 0],
 [ 0, 0,34,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,29, 0, 0],
 [ 0, 0, 0,..., 0,43, 0],
 [ 0, 0, 0,..., 0, 0,34]]

RandomForest Accuarcy 1.0
RandomForest F1 Score1.0
RandomForest Confusion Matrix:
[[34, 0, 0,..., 0, 0, 0],
 [ 0,41, 0,..., 0, 0, 0],
 [ 0, 0,34,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,29, 0, 0],
 [ 0, 0, 0,..., 0,43, 0],
 [ 0, 0, 0,..., 0, 0,34]]

Multinomial Accuarcy 1.0
Multinomial F1 Score1.0
Multinomial Confusion Matrix:
[[34, 0, 0,..., 0, 0, 0],
 [ 0,41, 0,..., 0, 0, 0],
 [ 0, 0,34,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,29, 0, 0],
 [ 0, 0, 0,..., 0,43, 0],
 [ 0, 0, 0,..., 0, 0,34]]

KNearestNeighbors Accuarcy 1.0
KNearestNeighbors F1 Score1.0
```

```
KNearestNeighbors Confusion Matrix:
[[34, 0, 0,..., 0, 0, 0],
 [ 0,41, 0,..., 0, 0, 0],
 [ 0, 0,34,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,29, 0, 0],
 [ 0, 0, 0,..., 0,43, 0],
 [ 0, 0, 0,..., 0, 0,34]]
```

Chosed Random Forest Classifier among all the models

In [16]:
```
rfc = RandomForestClassifier(n_estimators=120, random_state=42)
rfc.fit(X_train, y_train)
predictions = rfc.predict(X_test)
accuracy_score(y_test, predictions)
```

Out[16]: 1.0

In [17]:
```
rfc
```

Out[17]:

RandomForestClassifier

```
RandomForestClassifier(n_estimators=120, random_state=42)
```

Saving the model

In [18]:
```
import pickle
pickle.dump(rfc, open("rfc.pkl","wb"))
```

Loading the model for prediction on the basis of user input

In [19]:
```
rfc = pickle.load(open("rfc.pkl","rb"))
```

In [20]:
```
input1 = X_test.iloc[0].values.reshape(1,-1)
```

In [21]:
```
print("Predicted Label: ", rfc.predict(input1))
print("Actual Label: ", y_test[0])
```

```
Predicted Label:  [3]
Actual Label:  3
```

Loading the supportive datasets to give the complete solution on predicted disease

In [22]:
```
Symptoms_disease = pd.read_csv("symtoms_df.csv")
precautions = pd.read_csv("precautions_df.csv")
workout = pd.read_csv("workout_df.csv")
description = pd.read_csv("description.csv")
medication = pd.read_csv("medications.csv")
diet = pd.read_csv("diets.csv")
```

In [23]:
```
symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering': 4, 'chills': 5, 'joint_pai
diseases_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction', 33: 'Peptic ulcer diseae'
```

Please select the number corresponding to the symptoms you're experiencing to determine the associated disease.

In [24]:
```
def get_predicted_value(patient_symptoms):
    input_vector = np.zeros(len(symptoms_dict))
    for item in patient_symptoms:
        input_vector[symptoms_dict[item]] = 1
    return diseases_list[rfc.predict([input_vector])[0]]

print("Select symptoms by entering their corresponding numbers separated by commas:")
for index, symptom in symptoms_dict.items():
    print(f"{symptom}: {index}")

selected_symptoms_input = input("Enter the numbers of the symptoms you're experiencing (separated by commas): ")

selected_symptom_nums = selected_symptoms_input.split(',')

selected_symptoms = []

for num in selected_symptom_nums:
    if num.strip().isdigit() and int(num.strip()) in range(len(symptoms_dict)):
        selected_symptoms.append(list(symptoms_dict.keys())[int(num.strip())])
    else:
        print(f"Invalid input: {num}. Skipping...")

print("Selected symptoms:")
for symptom in selected_symptoms:
    print(symptom)

predicted_disease = get_predicted_value(selected_symptoms)
```

```
print("Predicted Disease:", predicted_disease)
```

```
Select symptoms by entering their corresponding numbers separated by commas:
0: itching
1: skin_rash
2: nodal_skin_eruptions
3: continuous_sneezing
4: shivering
5: chills
6: joint_pain
7: stomach_pain
8: acidity
9: ulcers_on_tongue
10: muscle_wasting
11: vomiting
12: burning_micturition
13: spotting_ urination
14: fatigue
15: weight_gain
16: anxiety
17: cold_hands_and_feets
18: mood_swings
19: weight_loss
20: restlessness
21: lethargy
22: patches_in_throat
23: irregular_sugar_level
24: cough
25: high_fever
26: sunken_eyes
27: breathlessness
28: sweating
29: dehydration
30: indigestion
31: headache
32: yellowish_skin
33: dark_urine
34: nausea
35: loss_of_appetite
36: pain_behind_the_eyes
37: back_pain
38: constipation
39: abdominal_pain
40: diarrhoea
41: mild_fever
42: yellow_urine
43: yellowing_of_eyes
44: acute_liver_failure
```

```
45: fluid_overload
46: swelling_of_stomach
47: swelled_lymph_nodes
48: malaise
49: blurred_and_distorted_vision
50: phlegm
51: throat_irritation
52: redness_of_eyes
53: sinus_pressure
54: runny_nose
55: congestion
56: chest_pain
57: weakness_in_limbs
58: fast_heart_rate
59: pain_during_bowel_movements
60: pain_in_anal_region
61: bloody_stool
62: irritation_in_anus
63: neck_pain
64: dizziness
65: cramps
66: bruising
67: obesity
68: swollen_legs
69: swollen_blood_vessels
70: puffy_face_and_eyes
71: enlarged_thyroid
72: brittle_nails
73: swollen_extremeties
74: excessive_hunger
75: extra_marital_contacts
76: drying_and_tingling_lips
77: slurred_speech
78: knee_pain
79: hip_joint_pain
80: muscle_weakness
81: stiff_neck
82: swelling_joints
83: movement_stiffness
84: spinning_movements
85: loss_of_balance
86: unsteadiness
87: weakness_of_one_body_side
88: loss_of_smell
89: bladder_discomfort
90: foul_smell_of urine
```

```
91: continuous_feel_of_urine
92: passage_of_gases
93: internal_itching
94: toxic_look_(typhos)
95: depression
96: irritability
97: muscle_pain
98: altered_sensorium
99: red_spots_over_body
100: belly_pain
101: abnormal_menstruation
102: dischromic _patches
103: watering_from_eyes
104: increased_appetite
105: polyuria
106: family_history
107: mucoid_sputum
108: rusty_sputum
109: lack_of_concentration
110: visual_disturbances
111: receiving_blood_transfusion
112: receiving_unsterile_injections
113: coma
114: stomach_bleeding
115: distention_of_abdomen
116: history_of_alcohol_consumption
117: fluid_overload.1
118: blood_in_sputum
119: prominent_veins_on_calf
120: palpitations
121: painful_walking
122: pus_filled_pimples
123: blackheads
124: scurring
125: skin_peeling
126: silver_like_dusting
127: small_dents_in_nails
128: inflammatory_nails
129: blister
130: red_sore_around_nose
131: yellow_crust_ooze
Selected symptoms:
continuous_sneezing
shivering
Predicted Disease: Allergy
```

In [25]:
```python
def helper(dis):
    # Get disease description
    desc = description[description['Disease'] == dis]['Description']
    desc = " ".join([w for w in desc])

    # Get precautions
    pre = precautions[precautions['Disease'] == dis][['Precaution_1', 'Precaution_2', 'Precaution_3', 'Precaution_4']]
    pre = [col for col in pre.values]

    # Get medication
    med = medication[medication['Disease'] == dis]['Medication']
    med = [med for med in med.values]

    # Get diet
    die = diet[diet['Disease'] == dis]['Diet']
    die = [die for die in die.values]

    # Get workout
    wrkout = workout[workout['disease'] == dis]['workout']

    return desc, pre, med, die, wrkout

desc, pre, med, die, wrkout = helper(predicted_disease)
```

Here's the full solution for your illness. Take care!

In [28]:
```python
desc, pre, med, die, wrkout = helper(predicted_disease)

print("=================predicted disease============")
print(predicted_disease, "\n")
print("=================description==================")
print(desc, "\n")
print("=================precautions==================")
i = 1
j = 1
k = 1
l = 1
for p_i in pre[0]:
    print(i, ": ", p_i)
    i += 1
print()
print("=================medications==================")
for m_i in med:
    print(j, ": ", m_i)
```

```
    j += 1
print()
print("================workout=================")
for w_i in wrkout:
    print(k, ": ", w_i)
    k += 1
print()
print("================diets=================")
for d_i in die:
    print(l, ": ", d_i)
    l += 1
```

```
================predicted disease============
Allergy

================description=================
Allergy is an immune system reaction to a substance in the environment.

================precautions=================
1 :  apply calamine
2 :  cover area with bandage
3 :  nan
4 :  use ice to compress itching

================medications=================
1 :  ['Antihistamines', 'Decongestants', 'Epinephrine', 'Corticosteroids', 'Immunotherapy']

================workout=================
1 :  Avoid allergenic foods
2 :  Consume anti-inflammatory foods
3 :  Include omega-3 fatty acids
4 :  Stay hydrated
5 :  Eat foods rich in vitamin C
6 :  Include quercetin-rich foods
7 :  Consume local honey
8 :  Limit processed foods
9 :  Include ginger in diet
10 :  Avoid artificial additives

================diets=================
1 :  ['Elimination Diet', 'Omega-3-rich foods', 'Vitamin C-rich foods', 'Quercetin-rich foods', 'Probiotics']
```

THANK YOU