In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Bhawana gupta\Downloads\medicine recommendation dataset\Training.csv")
```

In [3]:
```
df
```

Out[3]:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	..
0	1	1	1	0	0	0	0	0	0	0	..
1	0	1	1	0	0	0	0	0	0	0	..
2	1	0	1	0	0	0	0	0	0	0	..
3	1	1	0	0	0	0	0	0	0	0	..
4	1	1	1	0	0	0	0	0	0	0	..
...	...	...	...	...	...	...	...	...	...	...	..
4915	0	0	0	0	0	0	0	0	0	0	..
4916	0	1	0	0	0	0	0	0	0	0	..
4917	0	0	0	0	0	0	0	0	0	0	..
4918	0	1	0	0	0	0	1	0	0	0	..
4919	0	1	0	0	0	0	0	0	0	0	..

4920 rows × 133 columns

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

Out[4]:

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

5 rows × 133 columns

In [5]:
```
df.shape
```

Out[5]:
```
(4920, 133)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4920 entries, 0 to 4919
Columns: 133 entries, itching to prognosis
dtypes: int64(132), object(1)
memory usage: 5.0+ MB
```

In [7]:
```
len(df['prognosis'].unique())
```

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

In [8]:
```
df['prognosis'].unique()
```

Out[8]:
```
array(['Fungal infection', 'Allergy', 'GERD', 'Chronic cholestasis',
       'Drug Reaction', 'Peptic ulcer diseae', 'AIDS', 'Diabetes ',
       'Gastroenteritis', 'Bronchial Asthma', 'Hypertension ', 'Migraine',
       'Cervical spondylosis', 'Paralysis (brain hemorrhage)', 'Jaundice',
       'Malaria', 'Chicken pox', 'Dengue', 'Typhoid', 'hepatitis A',
       'Hepatitis B', 'Hepatitis C', 'Hepatitis D', 'Hepatitis E',
       'Alcoholic hepatitis', 'Tuberculosis', 'Common Cold', 'Pneumonia',
       'Dimorphic hemmorhoids(piles)', 'Heart attack', 'Varicose veins',
       'Hypothyroidism', 'Hyperthyroidism', 'Hypoglycemia',
       'Osteoarthristis', 'Arthritis',
       '(vertigo) Paroymsal  Positional Vertigo', 'Acne',
       'Urinary tract infection', 'Psoriasis', 'Impetigo'], dtype=object)
```

In [9]:
```
# train test split
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

In [10]:
```
X = df.drop('prognosis',axis=1)
y = df['prognosis']
```

In [11]:
```
# Transforming string values into numeric data
print(y)
le = LabelEncoder()
le.fit(y)
Y = le.transform(y)
print(Y)
```

```
0                              Fungal infection
1                              Fungal infection
2                              Fungal infection
3                              Fungal infection
4                              Fungal infection
                          ...
4915    (vertigo) Paroymsal  Positional Vertigo
4916                                       Acne
4917                    Urinary tract infection
4918                                  Psoriasis
4919                                   Impetigo
Name: prognosis, Length: 4920, dtype: object
[15 15 15 ... 38 35 27]
```

In [12]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=20)
# Ensure data is in NumPy array format
X_train = X_train.values
X_test = X_test.values
```

In [13]:
```
X_train.shape,X_test.shape,Y_train.shape,Y_test.shape
```

Out[13]:
```
((3444, 132), (1476, 132), (3444,), (1476,))
```

In [14]:
```
# Training top models
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
# For Checking accuracy
from sklearn.metrics import accuracy_score,confusion_matrix
import numpy as np

# Create a dictionary to store models
models = {
    "SVC":SVC(kernel='linear'),
    "RandomForest":RandomForestClassifier(n_estimators=100,random_state=42),
    "GradientBoosting":GradientBoostingClassifier(n_estimators=100,random_state=42),
    "KNeighbors":KNeighborsClassifier(n_neighbors=5),
    "MultinomialNB":MultinomialNB()
}
for model_name,model in models.items():
    # train model
    model.fit(X_train,Y_train)
    #test model
    predictions = model.predict(X_test)
    # calculate accuracy
    accuracy=accuracy_score(Y_test,predictions)
    # calculate confusion matrix
    cm = confusion_matrix(Y_test,predictions)
    print(f"{model_name} accuracy : {accuracy}")
    print(f"{model_name} confusion matrix:")
    print(np.array2string(cm,separator=','))
```

```
SVC accuracy : 1.0
SVC confusion matrix:
[[40, 0, 0,..., 0, 0, 0],
 [ 0,43, 0,..., 0, 0, 0],
 [ 0, 0,28,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,34, 0, 0],
 [ 0, 0, 0,..., 0,41, 0],
 [ 0, 0, 0,..., 0, 0,31]]
RandomForest accuracy : 1.0
RandomForest confusion matrix:
[[40, 0, 0,..., 0, 0, 0],
 [ 0,43, 0,..., 0, 0, 0],
 [ 0, 0,28,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,34, 0, 0],
 [ 0, 0, 0,..., 0,41, 0],
 [ 0, 0, 0,..., 0, 0,31]]
GradientBoosting accuracy : 1.0
GradientBoosting confusion matrix:
[[40, 0, 0,..., 0, 0, 0],
 [ 0,43, 0,..., 0, 0, 0],
 [ 0, 0,28,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,34, 0, 0],
 [ 0, 0, 0,..., 0,41, 0],
 [ 0, 0, 0,..., 0, 0,31]]
KNeighbors accuracy : 1.0
KNeighbors confusion matrix:
[[40, 0, 0,..., 0, 0, 0],
 [ 0,43, 0,..., 0, 0, 0],
 [ 0, 0,28,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,34, 0, 0],
 [ 0, 0, 0,..., 0,41, 0],
 [ 0, 0, 0,..., 0, 0,31]]
MultinomialNB accuracy : 1.0
MultinomialNB confusion matrix:
[[40, 0, 0,..., 0, 0, 0],
 [ 0,43, 0,..., 0, 0, 0],
 [ 0, 0,28,..., 0, 0, 0],
 ...,
 [ 0, 0, 0,...,34, 0, 0],
 [ 0, 0, 0,..., 0,41, 0],
 [ 0, 0, 0,..., 0, 0,31]]
```

In [15]:
```
# Single prediction
svc = SVC(kernel='linear')
svc.fit(X_train,Y_train)
ypred=svc.predict(X_test)
accuracy_score(Y_test,ypred)
```

Out[15]:
```
1.0
```

In [16]:
```
# saving model
import pickle
pickle.dump(svc,open("model/svc.pkl",'wb'))
```

In [17]:
```
# Load the model
svc = pickle.load(open("model/svc.pkl",'rb'))
```

In [18]:
```
# Test 1
print("Predicted Label :",svc.predict(X_test[0].reshape(1,-1)))
print("Actual Label:",Y_test[0])
```

```
Predicted Label : [40]
Actual Label: 40
```

In [19]:
```
# Test 1
print("Predicted Label :",svc.predict(X_test[10].reshape(1,-1)))
print("Actual Label:",Y_test[10])
```

```
Predicted Label : [20]
Actual Label: 20
```

In [20]:
```
# Load Database and use logic for recommendations
sym_des = pd.read_csv(r"C:\Users\Bhawana gupta\Downloads\medicine recommendation dataset\symtoms_df.csv")
precautions = pd.read_csv(r"C:\Users\Bhawana gupta\Downloads\medicine recommendation dataset\precautions_df.csv
workout = pd.read_csv(r"C:\Users\Bhawana gupta\Downloads\medicine recommendation dataset\workout_df.csv")
description = pd.read_csv(r"C:\Users\Bhawana gupta\Downloads\medicine recommendation dataset\description.csv")
medications = pd.read_csv(r"C:\Users\Bhawana gupta\Downloads\medicine recommendation dataset\medications.csv")
diets = pd.read_csv(r"C:\Users\Bhawana gupta\Downloads\medicine recommendation dataset\diets.csv")
```

In [21]:
```
np.zeros(10)
```

Out[21]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [22]:
```
symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering'
```

In [22]:
```
symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering'
diseases_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction

# model prediction function
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[svc.predict([input_vector])[0]]
```

In [24]:
```
# test 1
symptoms = input("Enter your symptoms.. ")
user_symptoms = [s.strip() for s in symptoms.split(',')]
user_symptoms = [sym.strip("[]'") for sym in user_symptoms]
predicted_disease = get_predicted_value(user_symptoms)
predicted_disease
```

```
Enter your symptoms.. shivering,chills,joint_pain,stomach_pain,acidity
```

Out[24]:
```
'Allergy'
```

In [26]:
```
def helper(dis):
    desc = description[description['Disease']==dis]['Description']
    desc =" ".join([w for w in desc])
    pre = precautions[precautions['Disease']==dis][['Precaution_1','Precaution_2','Precaution_3','Precaution_4'
    pre = [col for col in pre.values]
    med = medications[medications['Disease']==dis]['Medication']
    med = [med for med in med.values]
    die = diets[diets['Disease'] == dis]['Diet']
    die = [die for die in die.values]
    wrkout = workout[workout['disease']==dis]['workout']

    return desc,pre,med,die,wrkout
symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering'
diseases_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction

# model prediction function
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[svc.predict([input_vector])[0]]
```

In [27]:
```
# test 1:
symptoms = input("Enter your symptoms..")
user_symptoms = [s.strip() for s in symptoms.split(',')]
user_symptoms = [sym.strip("[]' ") for sym in user_symptoms]
```

```
Enter your symptoms..shivering,chills,joint_pain,stomach_pain,acidity
```

In [28]:
```
predicted_diseases = get_predicted_value(user_symptoms)
predicted_diseases
desc,pre,med,die,wrkout = helper(predicted_diseases)
```

In [29]:
```
print(predicted_diseases)
print(desc)
print(pre)
print(med)
print(die)
print(wrkout)
```

```
Allergy
Allergy is an immune system reaction to a substance in the environment.
[array(['apply calamine', 'cover area with bandage', nan,
       'use ice to compress itching'], dtype=object)]
["['Antihistamines', 'Decongestants', 'Epinephrine', 'Corticosteroids', 'Immunotherapy']"]
["['Elimination Diet', 'Omega-3-rich foods', 'Vitamin C-rich foods', 'Quercetin-rich foods', 'Probiotics']"]
10             Avoid allergenic foods
11    Consume anti-inflammatory foods
12        Include omega-3 fatty acids
13                       Stay hydrated
14        Eat foods rich in vitamin C
15       Include quercetin-rich foods
16                Consume local honey
17              Limit processed foods
18             Include ginger in diet
19         Avoid artificial additives
Name: workout, dtype: object
```

In [36]:
```
import sklearn
print(sklearn.__version__)
```

```
1.3.2
```

In [132…
```
pip install scikit-learn==1.3.2
```

```
Collecting scikit-learn==1.3.2Note: you may need to restart the kernel to use updated packages.

  Obtaining dependency information for scikit-learn==1.3.2 from https://files.pythonhosted.org/packages/4e/ba/ce9bd1cd4953336a0e213b29cb80bb11816f2a93de8c99f88ef0b446ad0c/scikit_learn-1.3.2-cp311-cp311-win_amd64.whl.metadata
  Downloading scikit_learn-1.3.2-cp311-cp311-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\bhawana gupta\anaconda3\lib\site-packages (from scikit-learn==1.3.2) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\bhawana gupta\anaconda3\lib\site-packages (from scikit-learn==1.3.2) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\bhawana gupta\anaconda3\lib\site-packages (from scikit-learn==1.3.2) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\bhawana gupta\anaconda3\lib\site-packages (from scikit-learn==1.3.2) (2.2.0)
Downloading scikit_learn-1.3.2-cp311-cp311-win_amd64.whl (9.2 MB)
   ---------------------------------------- 0.0/9.2 MB ? eta -:--:--
   - -------------------------------------- 0.4/9.2 MB 7.4 MB/s eta 0:00:02
   --- ------------------------------------ 0.8/9.2 MB 8.7 MB/s eta 0:00:01
   ----- ---------------------------------- 1.4/9.2 MB 9.5 MB/s eta 0:00:01
   -------- ------------------------------- 1.9/9.2 MB 9.4 MB/s eta 0:00:01
   ---------- ----------------------------- 2.4/9.2 MB 10.1 MB/s eta 0:00:01
   ------------ --------------------------- 3.0/9.2 MB 10.6 MB/s eta 0:00:01
   --------------- ------------------------ 3.5/9.2 MB 10.7 MB/s eta 0:00:01
   ---------------- ----------------------- 3.8/9.2 MB 10.2 MB/s eta 0:00:01
   ------------------ --------------------- 4.4/9.2 MB 10.0 MB/s eta 0:00:01
   -------------------- ------------------- 4.6/9.2 MB 9.8 MB/s eta 0:00:01
   -------------------- ------------------- 4.8/9.2 MB 9.0 MB/s eta 0:00:01
   --------------------- ------------------ 5.0/9.2 MB 8.8 MB/s eta 0:00:01
   ---------------------- ----------------- 5.3/9.2 MB 8.6 MB/s eta 0:00:01
   ------------------------ --------------- 5.6/9.2 MB 8.3 MB/s eta 0:00:01
   ------------------------- -------------- 5.8/9.2 MB 8.3 MB/s eta 0:00:01
   ------------------------- -------------- 5.9/9.2 MB 7.7 MB/s eta 0:00:01
   -------------------------- ------------- 6.0/9.2 MB 7.5 MB/s eta 0:00:01
   -------------------------- ------------- 6.1/9.2 MB 7.2 MB/s eta 0:00:01
   --------------------------- ------------ 6.2/9.2 MB 6.9 MB/s eta 0:00:01
   --------------------------- ------------ 6.2/9.2 MB 6.6 MB/s eta 0:00:01
   ---------------------------- ----------- 6.2/9.2 MB 6.2 MB/s eta 0:00:01
   ---------------------------- ----------- 6.3/9.2 MB 6.0 MB/s eta 0:00:01
   ----------------------------- ---------- 6.4/9.2 MB 5.9 MB/s eta 0:00:01
   ----------------------------- ---------- 6.6/9.2 MB 5.7 MB/s eta 0:00:01
   ------------------------------ --------- 6.8/9.2 MB 5.7 MB/s eta 0:00:01
   ------------------------------ --------- 7.1/9.2 MB 5.7 MB/s eta 0:00:01
   ------------------------------- -------- 7.4/9.2 MB 5.7 MB/s eta 0:00:01
   ------------------------------- -------- 7.8/9.2 MB 5.8 MB/s eta 0:00:01
   -------------------------------- ------- 8.2/9.2 MB 5.9 MB/s eta 0:00:01
   -------------------------------- ------- 8.6/9.2 MB 5.9 MB/s eta 0:00:01
   --------------------------------- ------ 8.9/9.2 MB 6.1 MB/s eta 0:00:01
   ---------------------------------------  9.2/9.2 MB 6.0 MB/s eta 0:00:01
   ---------------------------------------- 9.2/9.2 MB 5.8 MB/s eta 0:00:00
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.3.0
    Uninstalling scikit-learn-1.3.0:
      Successfully uninstalled scikit-learn-1.3.0
Successfully installed scikit-learn-1.3.2
```

In [ ]: